WINDOWS MEMORY FORENSIC DATA VISUALIZATION

THESIS

J. Brendan Baum, Civilian, USAF

AFIT-ENG-T-14-J-1

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-T-14-J-1

WINDOWS MEMORY FORENSIC DATA VISUALIZATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

J. Brendan Baum, B.S.

Civilian, USAF

June 2014

AFIT-ENG-T-14-J-1

WINDOWS MEMORY FORENSIC DATA VISUALIZATION

J. Brendan Baum, B.S.

Civilian, USAF

Approved:

//signed//	28 May 2014
Gilbert L. Peterson, PhD (Chairman)	Date
//signed//	28 May 2014
Barry E. Mullins, PhD (Member)	Date
//signed//	28 May 2014
Timothy H. Lacey, PhD (Member)	Date

AFIT-ENG-T-14-J-1

Abstract

Modern criminal investigators face an increasing number of computer-related crimes that require the application of digital forensic science. The major challenge facing digital forensics practitioners is the complicated task of acquiring an understanding of the digital data residing in electronic devices. Currently, this task requires significant experience and background to correctly aggregate the data their tools provide from the digital artifacts. Most of the tools available present their results in text files or tree lists. It is up to the practitioner to mentally capture a global understanding of the state of the device at the time of seizure and find the items of evidentiary interest. This research focuses on the application of Information Visualization techniques to improve the analysis of digital forensic evidence from Microsoft Windows memory captures. The visualization tool developed in this work presents both global and local views of the evidence based on user interactions with the graphics. The resulting visualizations provide the necessary details for verifying digital artifacts and assists in locating additional items of relevance. This proof-of-concept model can be modified to support various digital forensic target platforms including Mac OS X, Linux, and Android.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Gilbert Peterson, for providing the idea for this thesis and for his guidance and support throughout. It would not have been possible to complete this research without his assistance and dedicated involvement throughout the entire process.

J. Brendan Baum

Table of Contents

Page

List of Figures

Figure Page

Figure Page

List of Tables

WINDOWS MEMORY FORENSIC DATA VISUALIZATION

I. Introduction

Criminal investigations in the modern era frequently involve digital forensic science to analyze potential digital evidence obtained from electronic devices. The Federal Bureau of Investigation (FBI) regularly investigates offenses such as Computer and Network Intrusions, Identity Theft, as well as cases involving Child Pornography (FBI, 2014). As computer technology advances, devices continue to increase their storage capacity and processing power effectively enlarging digital forensic evidence collections. Research by Beebe and Clark (2005) calls for improvement to the digital investigation process citing the rising temporal factor associated with digital evidence analysis. Data mining techniques involving predictive modeling and content retrieval are discussed as potential enhancements to the analysis process. Data mining employs methods from various fields including artificial intelligence, machine learning, pattern recognition, and data visualization (Beebe & Clark, 2005).

The applied science of Information Visualization offers unique solutions for gaining an intuitive understanding of large and complex datasets. Information Visualization (InfoVis) methods have led to innovative problem solving in numerous fields including medicine (Faisal, et al, 2013), business, and computer science (Liu, et al, 2014). Such methods could provide a fresh approach for Digital Forensic Analysts tackling the complexity of understanding large datasets.

Keim (2002) points out the significance of involving human interaction in the data exploration process by blending "the flexibility, creativity, and general knowledge of the

human with the enormous storage capacity and the computational power of today's computers." An advantage to employing a visual approach to analysis lies in the fact that exploration is instinctive. The human user (analyst) does not need to grasp complex mathematical or statistical concepts to successfully locate data of interest (Keim, 2002).

1.1 Problem Overview

The constant increase in electronic device storage capacity results in greater workloads for forensic analysts which expands the target space containing relevant digital evidence. Consequently the time required to properly analyze the collected data, identify relevant digital artifacts, and extract digital evidence increases exponentially. In summary, investigators face an uphill battle due to the substantial increase in the size of individual digital forensic data collections. The added complexity increases the time required to complete forensic analysis using traditional methods.

Although the majority of investigatory efforts focus on the data in disk storage, information recovered from volatile memory can provide unique insight into the state of the device at the time of acquisition (Cai, et al, 2013). Data regarding open ports and sockets provides an examiner with the state of network communications on the target system. The listing of open file handles and system registry keys relates the various resources in disk storage accessed by each system process. Knowledge of the resources associated with a particular process can assist in identifying key data being exfiltrated by malware on an infected system.

1.2 Research Motivation

The motivation behind this research stems from the growing need to develop innovative methods to aid in digital forensic analysis. One proposed method involves the application of Information Visualization techniques to provide a unique perspective on collection datasets (Osborne, 2012). According to Osborne, InfoVis tools have the potential to "aid in the discovery of new pieces of information, decision-making based on data, and to provide explanation of certain phenomena existent within the data set."

1.3 Research Objectives

The overall goal of this research is to develop a proof-of-concept interactive visualization for analysis of forensic memory images. Shneiderman (1996) outlines seven tasks for developing Information Visualization tools listed in Table 1.

Table 1. Seven Tasks for Information Visualization (Shneiderman, 1996).

Task	Description
Overview	Gain an overview of the entire collection.
Zoom	Zoom in on items of interest.
Filter	Filter out uninteresting items.
Details-on-Demand	Select an item or group and get details when needed.
Relate	View relationships among items.
History	Keep a history of actions to support undo, replay, and progressive refinement.
Extract	Allow extraction of sub-collections and of the query parameters.

The specific objectives for this research are derived from this list and outlined as follows:

1. The visualization tool should provide an interactive global view of the data output from forensic memory image analysis tools.
2. The visualization tool should have the ability to filter the datasets for a particular system and visually represent the following associations:
    - The relationship between each process and its associated resources.
    - The relationship between a system resource and its associated processes.
3. The visualization tool should assist the user in locating unique patterns and information of relevance within the datasets.

1.4 Methodology

Chapter 3 presents the process for obtaining forensic memory captures from Windows machines and the methodology for extracting digital artifacts from forensic memory captures. The two types of malware implemented on the test machines are discussed on a behavioral level. The second part of the chapter presents a functional overview of the developed visualization tool interface. The basic features of the software are discussed which include the global system view and the local view of the system which is obtained through user interactions. The last feature of the visualization tool involves the addition of visual links to connect resources and associated processes to assist the user in locating unique patterns and information of significance within the digital forensic dataset.

Chapter 4 describes the application of the developed visualization tool to eight test datasets acquired from Windows XP forensic memory images. The images include

five trusted images and three images infected with a form of malware. The visual results from each of the datasets are analyzed and evaluated against the research objectives to verify whether each of the three goals is successfully met.

1.5 Research Implications

This thesis study is unique because there has been very little published regarding the application of Information Visualization techniques to Digital Forensic Analysis, particularly in the sub-field of Memory Forensics. The visualization tool developed through this research is shown to provide a level of assistance in the recognition of existing patterns and the discovery of significant items within the test datasets. Visualization could lead to an entirely fresh approach for Digital Forensic Investigations by empowering digital forensic practitioners with the ability to quickly perform analysis and gain an intuitive understanding of the target system at the time of seizure.

1.6 Summary

This chapter discusses the challenges facing digital forensic investigators concerning the rising time frames required to analyze collection datasets that continue to grow in size and complexity. The problem overview is stated including the importance of volatile memory analysis. The research motivation describes the need to develop innovative methods to assist investigators in the digital forensic analysis process. An Information Visualization approach is selected as a means of improving the digital forensic process. Three specific objectives for the research are listed and a basic methodology is outlined.

Chapter 2 provides an overview of the digital forensic investigation process and introduces the subfield of Forensic Memory Analysis. The chapter also presents the significance of memory analysis, image acquisition methods, and memory analysis tools. The subject of Information Visualization is explored as a possible solution for overcoming digital forensic challenges and available InfoVis tools are discussed.

Chapter 3 outlines the methodology for obtaining forensic memory captures from Windows RAM images and the process for extracting digital artifacts using available open-source tools. The chapter also provides a functional overview of the visualization tool interface developed in this research. Chapter 4 performs analysis using the visualization tool on eight test datasets obtained from Windows XP Random-Access Memory (RAM) captures. The results are evaluated against the three research objectives for verification.

The final chapter draws conclusions regarding the results of the research, and highlights the overall accomplishments of this study. Future research is proposed to improve upon the design implemented in the D3 JavaScript Visualization Tool.

II. Literature Review

Due to the recent rise of computer crime, Digital Forensic Science has become an important subject for Law Enforcement Investigators. This chapter presents the process of digital forensics and highlights the specific field of Forensic Memory Analysis. Techniques for collecting Windows memory dumps are discussed along with available open-source tools for digital memory artifact extraction. The research presented in this paper focuses on overcoming the challenges related to digital forensic analysis through the application of Information Visualization. Previous related research is discussed and a popular visualization tool is explored.

2.1 Digital Forensics

Cyber Crimes involving digital evidence are routinely investigated by law enforcement agencies. This field of analysis is commonly referred to as digital forensics, or digital forensic science. According to the United States Computer Emergency Readiness Team (US-CERT) digital forensics is defined as "the discipline that combines elements of law and computer science to collect and analyze data in a way that is admissible as evidence in a court of law" (US-CERT, 2008).

There are various subdivisions of digital forensics that are generally categorized under the following fields:

1. Computer Forensics – the preservation, identification, extraction, documentation and interpretation of computer data (Kruse & Heiser, 2002).

2. Mobile Forensics – the analysis of digital evidence obtained from devices such as smartphones, cellphones, and PDAs (NIST, 2007).
3. Network Forensics – the analysis of data collected from active computer network traffic to assist with intrusion detection, auditing, and monitoring (Palmer, 2001).
4. Database Forensics – the study of databases and their associated metadata (Yadav, 2011).

A commonly held standard model of the digital forensics process comprising seven separate stages was developed by the Digital Forensic Research Workshop (Palmer, 2001). The U.S. Department of Justice (DOJ) currently follows the digital forensic model outlined below (Carroll, et al, 2008):

1. *Obtaining and Imaging Forensic Data*
2. *Forensic Request*
3. *Preparation/Extraction*
4. *Identification*
5. *Analysis*
6. *Forensic Reporting*
7. *Case Level Analysis*

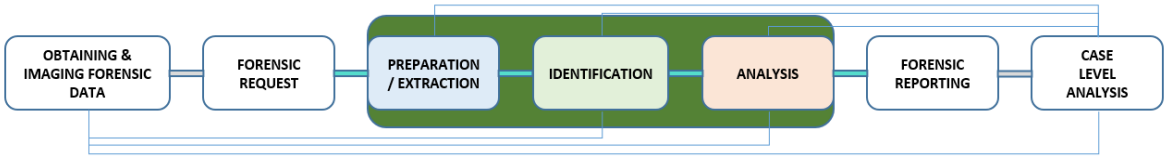

Figure 1. Digital Forensic Process (Carroll, et al, 2008).

The visualization method described in Chapter 3 relates to the Analysis phase of the DOJ digital forensics process. Before analysis can be performed, collection of appropriate data must occur. For this reason, the collection and examination of digital evidence is discussed here.

2.2 Sources of Digital Evidence

Digital evidence is defined by the Scientific Working Group on Digital Evidence (SWGDE) as "[i]nformation of probative value that is stored or transmitted in a binary form" (SWGDE, 2011). Computer hardware components which contain storage devices and have the potential to contain digital evidence include (NIJ, 2008):

- Hard Drives – Internal and External
- Removable Media – CDs, DVDs, Floppy Disks
- USB Thumb Drives
- Memory Cards – Secure Digital (SD) Cards, Compact Flash Cards
- Handheld Devices – Smartphones, PDAs, Tablets, Digital Media Devices (Cameras, mp3 Players, iPods, etc.)

Disk forensics collection methods applied to these devices can provide file-system data such as stored files and installed applications. Device physical memory can contain a wealth of information not found elsewhere (Carvey, 2009) but is omitted from this list. This is due to the volatile nature of memory where forensic collection requires careful measures to be taken in order to avoid altering the current state. Most incident response plans include the collection of volatile data (NIST, 2006).

2.3 Volatile Memory Forensics

Memory forensics involves analyzing a static memory image in order to determine the current state of the target system. This can be compared to taking a snapshot of the system memory for a particular instant in time. Live computer systems contain volatile data stored in Random-Access Memory (RAM). If power is no longer supplied to the system, the data stored in RAM is lost. Valuable information such as network connections, command history, and active process information resides solely in RAM (Carvey, 2009). Unique forensic artifacts relating to malware and rootkits reside in memory and cannot be discovered through hard disk forensic analysis (Davis, 2008). For this reason, the application of memory forensics techniques is critical to obtaining key digital evidence.

2.4 Memory Acquisition

Before digital evidence can be obtained from the volatile memory, forensic examiners must acquire a physical memory dump (image) from the target device. For the purposes of this research, all target machines studied utilize the Microsoft Windows XP operating system. Therefore this section discusses the available tools and methods for extracting the contents of physical memory from Windows XP computer systems. Methods of volatile memory acquisition are categorized as either hardware-based or software-based and are discussed in the following sections.

2.4.1 Hardware-Based Acquisition

A PCI expansion card called the Tribble was designed as a proof-of-concept device capable of accessing the contents of physical memory on a target computer

(Carrier, 2004). The acquisition is performed solely through hardware interaction without the need for software to be loaded onto the target. Unfortunately this method requires the Tribble card to be connected to the intended system prior to an event warranting forensic examination. For this reason, the Tribble is impractical for use in digital forensic investigations.

Currently the most popular hardware-based option for live memory acquisition on Windows XP systems involves the use of an external Firewire (IEEE 1394) device. This method essentially bypasses the operating system because Firewire specifications allow client devices to directly access the host system's memory (Burdach, 2006). An advantage to this technique is the high-speed data transfer rate (approximately 800 Mbit/sec) obtained through a Firewire connection.

2.4.2 Software-Based Acquisition

There are many software-based approaches to volatile data collection which are compatible with the Windows XP operating system. Similar to the /dev/mem device in Unix operating systems, Windows XP provides RAM access by means of an object device (\\.\PhysicalMemory). The Unix based data dumper tool (`dd`) and the various open source versions ported for Windows systems have the ability to read from these memory devices and output the data to a local image file (Garnet, 2013). An additional option of `dd` allows the output file to be written to a server on the network (Carrier, 2004). Other open source tools for volatile memory acquisition similar to `dd` include `Nigilant32`, `Win32dd`, and `Memoryze` (Carvey, 2009). `Win32dd` was selected for use in this research as described in Chapter 3.

The fundamental disadvantage of memory image collection using software-based approaches relates to the modifications to the system state due to the collection program interactions with the operating system. Software used to extract the image has to be loaded into memory, thereby modifying data currently stored in RAM (Davis, 2008). A similar issue relates to the operations performed by the kernel during the acquisition process. Processor activity that occurs parallel to the collection process affects the output image and potentially overwriting digital evidence artifacts (Carvey, 2009). Unfortunately, with a software-based approach utilizing the system kernel is essential due to processor scheduling and the movement of data to storage (Carrier, 2004).

2.5 Memory Dump Extraction Tools

The third phase of the DOJ digital forensics process model relates to the examination of collected data. With respect to memory forensics, this stage involves the application of forensic tools on raw memory dumps in an effort to extract digital evidence artifacts.

2.5.1 Volatility Framework

Volatility is an open source advanced memory forensics framework released under the GNU General Public License. The framework is written in the Python scripting language and designed to analyze RAM dumps and extract specific digital artifacts. A variety of operating systems are supported including Windows, Linux, Mac OS X, and Android. The modular design of Volatility allows for the support of future operating systems and architectures. The Volatility Project community is comprised of researchers

and professionals operating in the fields of malware analysis, incident response, and digital forensics (Volatility, 2013).

Members of the community develop individual module plugins for the framework and submit them to the project for general use. Plugins are organized into the following categories:

- Image Identification
- Processes and DLLs
- Process Memory
- Kernel Memory and Objects
- Networking
- Registry
- Crash Dumps, Hibernation, and Conversion
- File System
- Miscellaneous

Table 2 lists some of the most common modules designed for the Windows family of operating systems included in the most recent Volatility release (version 2.3.1). These are subsequently used in the research detailed in Chapter 3.

Table 2. Volatility (2.3.1) modules for Windows.

Module Plugin	Description
imageinfo	Display identified operating system, service pack, and architecture
connections	Lists the TCP connections active at the time of memory acquisition
handles	Lists the open handles for all processes
pslist	List the active processes of the system
sockets	Lists the listening network sockets for any protocol

Analysis of physical memory images requires the identification of kernel structures and their specific locations. For Windows operating systems this involves obtaining the appropriate symbol files from the Microsoft Symbol Server (Microsoft, 2014). These symbol files must then be converted to a format that can be interpreted by the Volatility framework. Volatility includes a bundle of prebuilt profiles for most Windows operating systems containing the appropriate symbol files relevant to the OS and particular architecture in a format the software can interpret (Volatility, 2013).

The current release of Volatility (2.3.1) executes from a Windows command prompt using a standalone executable (see Figure 2). Plugin modules are run by following the command format below:

"volatility-2.3.1.standalone .exe" vol.py –f [path to memory dump]

--profile=[OSprofile] [module]

The output can also be saved to a text file:

"volatility-2.3.1.standalone .exe" vol.py –f [path to memory dump]

--profile=[OSprofile] [module] > [path/filename].txt

```
C:\XP Dumps>"volatility-2.3.1.standalone .exe" vol.py -f XPSP2_IE.mem --profile=WinXPSP2x86 pslist
Volatility Foundation Volatility Framework 2.3.1
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------
0x825c8830 System                    4      0     60      398 ------      0
0x821a7020 smss.exe                544      4      3       19 ------      0 2014-04-29 14:30:42 UTC+0000
0x824ed020 csrss.exe               608    544     12      453      0      0 2014-04-29 14:30:44 UTC+0000
0x82056128 winlogon.exe            632    544     19      563      0      0 2014-04-29 14:30:44 UTC+0000
0x821e3020 services.exe            676    632     16      353      0      0 2014-04-29 14:30:44 UTC+0000
0x82342020 lsass.exe               688    632     26      368      0      0 2014-04-29 14:30:45 UTC+0000
0x82314988 vmacthlp.exe            848    676      1       25      0      0 2014-04-29 14:30:45 UTC+0000
0x821b4c18 svchost.exe             864    676     16      192      0      0 2014-04-29 14:30:45 UTC+0000
0x821b57e8 svchost.exe             948    676     10      290      0      0 2014-04-29 14:30:45 UTC+0000
0x8231a558 svchost.exe            1040    676     82     1747      0      0 2014-04-29 14:30:45 UTC+0000
0x8202a658 svchost.exe            1088    676      6       84      0      0 2014-04-29 14:30:45 UTC+0000
0x821b83c0 svchost.exe            1140    676     13      177      0      0 2014-04-29 14:30:45 UTC+0000
0x8232bda0 explorer.exe           1496   1460     16      603      0      0 2014-04-29 14:30:47 UTC+0000
0x820b2da0 spoolsv.exe            1608    676     10      138      0      0 2014-04-29 14:30:47 UTC+0000
0x824e4a78 rundll32.exe           1732   1496      4       76      0      0 2014-04-29 14:30:48 UTC+0000
0x82048da0 vmtoolsd.exe           1740   1496      6      265      0      0 2014-04-29 14:30:48 UTC+0000
0x8204cc10 ctfmon.exe             1748   1496      1       78      0      0 2014-04-29 14:30:48 UTC+0000
0x8204d980 msmsgs.exe             1756   1496      3      193      0      0 2014-04-29 14:30:48 UTC+0000
```

Figure 2. Volatility Command Prompt in Windows.

2.5.2 Compiled Memory Analysis Tool (CMAT)

The Compiled Memory Analysis Tool (CMAT) is an alternative open source utility capable of analyzing and extracting data from Windows RAM dump images. CMAT is developed using C++ in contrast to the Volatility Framework which uses Python. CMAT also requires access to the appropriate program database (PDB) symbol files from the Microsoft Symbol Server to identify the appropriate kernel structures and locations (Okolica & Peterson, 2011). Unlike Volatility, CMAT automatically obtains the correct symbol files from the Microsoft Symbol Server directly and does not require formatting.

CMAT is executed from a command prompt in Windows (see Figure 3) using the following command format:

cmat.exe [dumpfile name] -data [path for output files]

Output from CMAT is organized into six feature files as listed in Table 3.

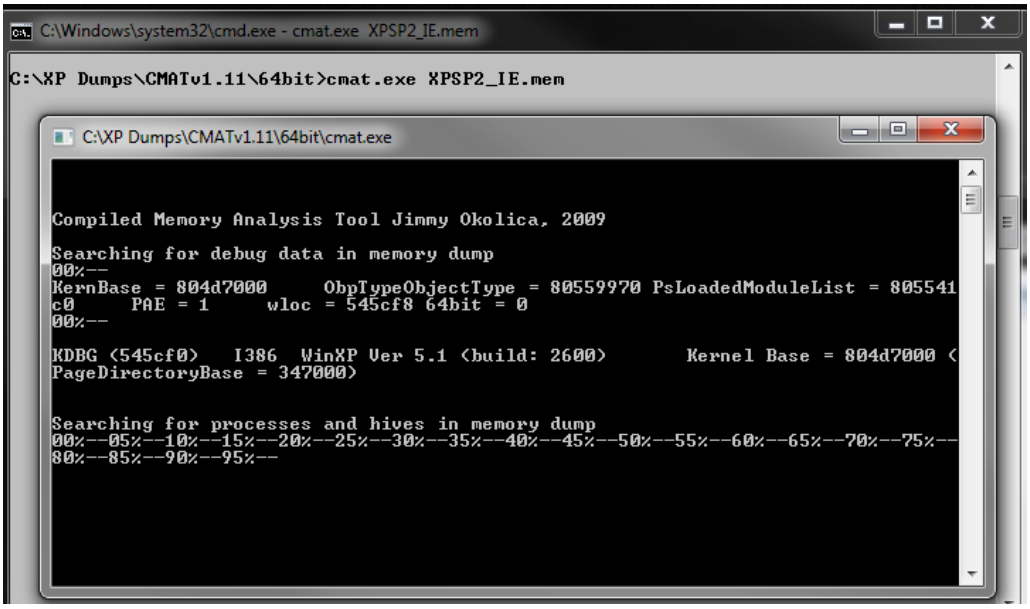

Figure 3. CMAT Command Prompt in Windows.

.

Table 3. CMAT Output Feature Files.

Feature File	Data	Details
1	Process Information	Process IDs, Process Names, User IDs
2	Network Information	Network connections active at time of memory acquisition
3	Process Loaded Modules	Loaded modules by Process ID
4	Process File Handles	Open Files by Process ID
5	Process Registry Keys	Registry Keys by Process ID
6	System Loaded Modules	System Drivers by Name

The feature files are placed in the specified output path. Figure 4 provides an example of the feature file contents.

```
180  svchost.exe     C:\WINDOWS\system32\svchost.exe
180  kernel32.dll    C:\WINDOWS\system32\kernel32.dll
180  ADVAPI32.dll    C:\WINDOWS\system32\ADVAPI32.dll
180  RPCRT4.dll      C:\WINDOWS\system32\RPCRT4.dll
180  Secur32.dll     C:\WINDOWS\system32\Secur32.dll
180  ShimEng.dll     C:\WINDOWS\system32\ShimEng.dll
180  AcGenral.DLL    C:\WINDOWS\AppPatch\AcGenral.DLL
180  USER32.dll      C:\WINDOWS\system32\USER32.dll
180  GDI32.dll       C:\WINDOWS\system32\GDI32.dll
180  WINMM.dll       C:\WINDOWS\system32\WINMM.dll
180  ole32.dll       C:\WINDOWS\system32\ole32.dll
180  msvcrt.dll      C:\WINDOWS\system32\msvcrt.dll
180  OLEAUT32.dll    C:\WINDOWS\system32\OLEAUT32.dll
180  MSACM32.dll     C:\WINDOWS\system32\MSACM32.dll
180  VERSION.dll     C:\WINDOWS\system32\VERSION.dll
```

Figure 4. CMAT Feature File Contents.

2.6 Forensic Data Analysis

The Analysis phase of the NIST digital forensic process occurs after all relevant data is extracted from the target device. Forensic Analysts tackle the difficult task of inspecting the various tables of extracted data in an attempt to gain a perspective on the state of the system at the time of memory acquisition. In many cases, the objective is to discover hidden activity or unusual phenomena in the extracted datasets.

2.6.1 Information Visualization

One proposed method of improving the Analysis process involves applying Information Visualization techniques to the forensic data. Humans tend to comprehend raw data more efficiently when it is presented in a visual form. Research has shown that people possess the skills to “visually interpret and comprehend pictures, video, and charts much faster than reading a textual description of the same” (Teerlink & Erbacher, 2006).

Shneiderman (1996) notes the human ability to "scan, recognize, and recall images rapidly" in addition to the natural capacity for identifying "changes in size, color, shape, movement, or texture."

Ware (2004), a leading researcher in the applied science of Information Visualization, lists the following advantages to utilizing InfoVis:

- Visualization provides an ability to comprehend huge amounts of data.
- Visualization allows the perception of emergent properties that were not anticipated.
- Visualization often enables problems with the data itself to become immediately apparent. A visualization commonly reveals things not only about the data itself, but about the way it is collected.
- Visualization facilitates understanding of both large-scale and small-scale features of the data.
- Visualization facilitates hypothesis formation.

2.6.2 EIC Process

The Explore, Investigate, and Correlate (EIC) process (Osborne, 2012) stems from recent research applying Information Visualization to digital forensics. The process consists of a set of methods to overcome the issue of comprehending large and complex digital forensic datasets.

The specific goals of the EIC process model are as follows:

- Make the evidence visible.
- Reduce the relative size of the evidence.

- Provide high-level overviews of the evidence first.
- Aid in the presentation of the events and relationships in the evidence.
- Provide explanations of the origin or significance of evidence.
- Provide support to help identify items of probative value.
- Facilitate the presentation of these findings to other investigators or in court.

In the Explore phase the human user is presented with a high-level overview of the data in its entirety. Through interactions with the visualization the user Investigates the data by applying filters or specifying a focus. The Correlate phase presents a low-level view which highlights specific relationships within the data as well as behavior patterns. Figure 5 provides a flow-chart diagram of the EIC Process. The research did not include the development of new visualization methods, but instead focused on applying established techniques that are simplistic in nature.

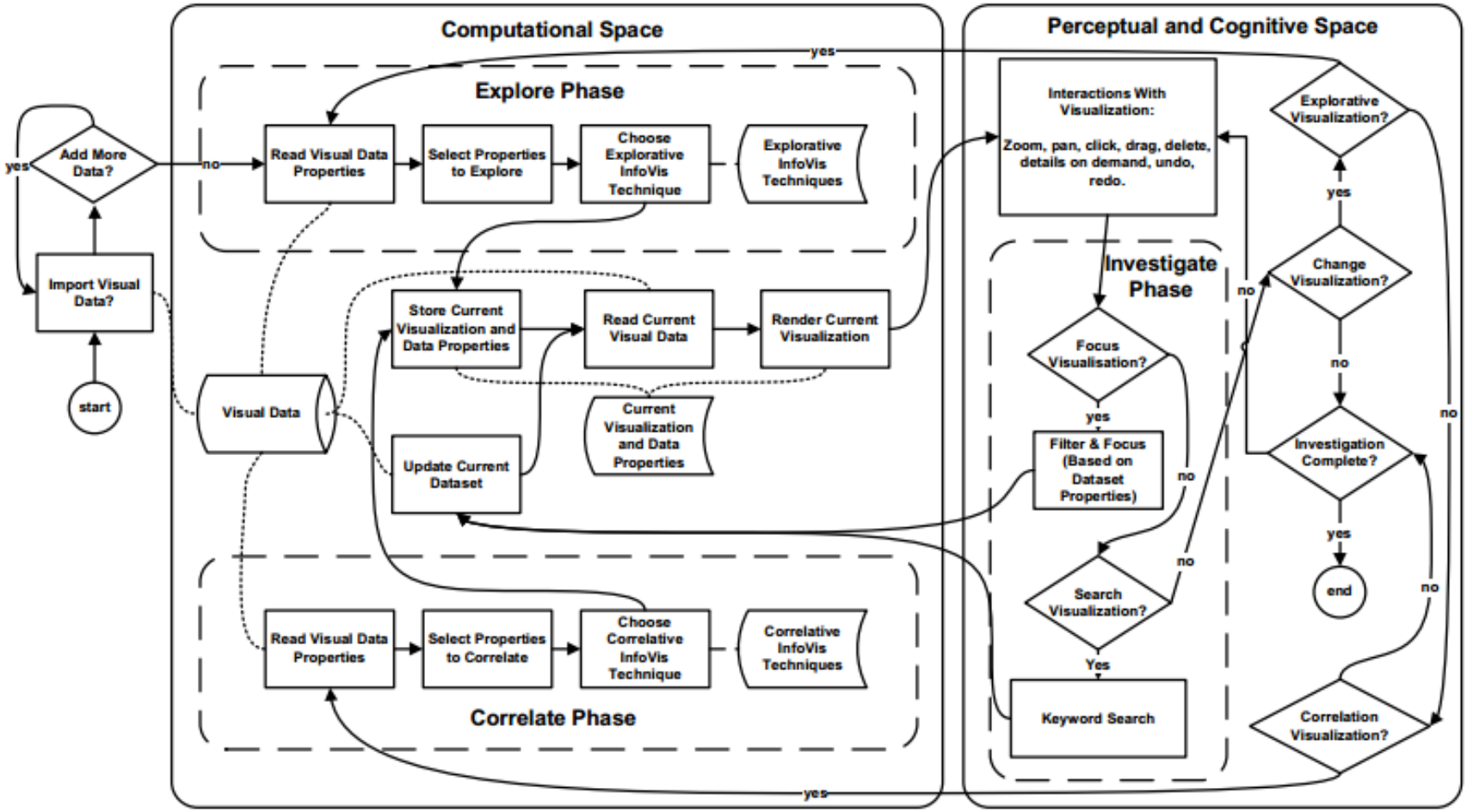

Figure 5. EIC Process Flowchart (Osborne, 2012).

2.7 Visualization Tools

Many tools exist for creating interactive visualization based on large datasets. Due to limited funding for this research the software packages explored in this section are limited to open-source libraries. Two of the most popular options are discussed here.

2.7.1 Gephi Visualization Software

The Gephi software package is capable of providing interactive graphical visualizations of complex datasets (Gephi, 2014). Gephi is especially helpful in presenting network data with various hierarchical and clustering characteristics (Bastian, et al, 2009). The software is written in Java and utilizes the OpenGL library for rendering graphical content. Figures 6 provides an example of a weighted network visualization using Gephi illustrating the co-appearance of characters in the novel *Les Miserables.*

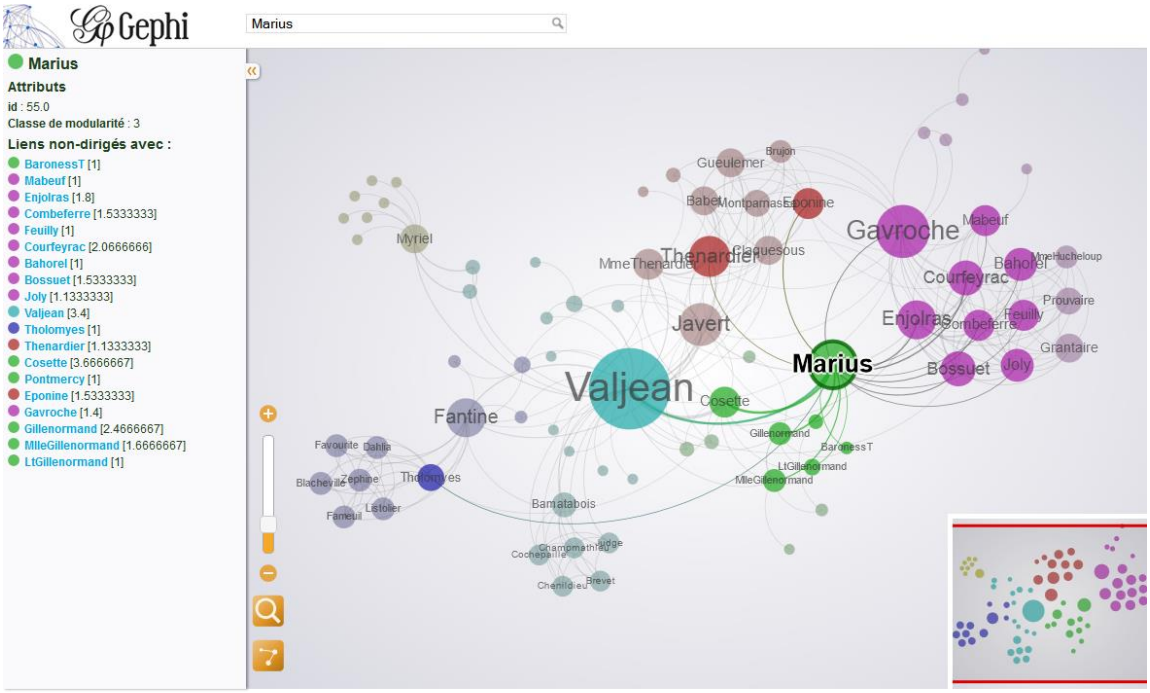

Figure 6. Co-Appearance of Characters in Les Miserables (Gephi, 2014).

Another example of Gephi visualization is shown in Figure 7 below which depicts Twitter network data.

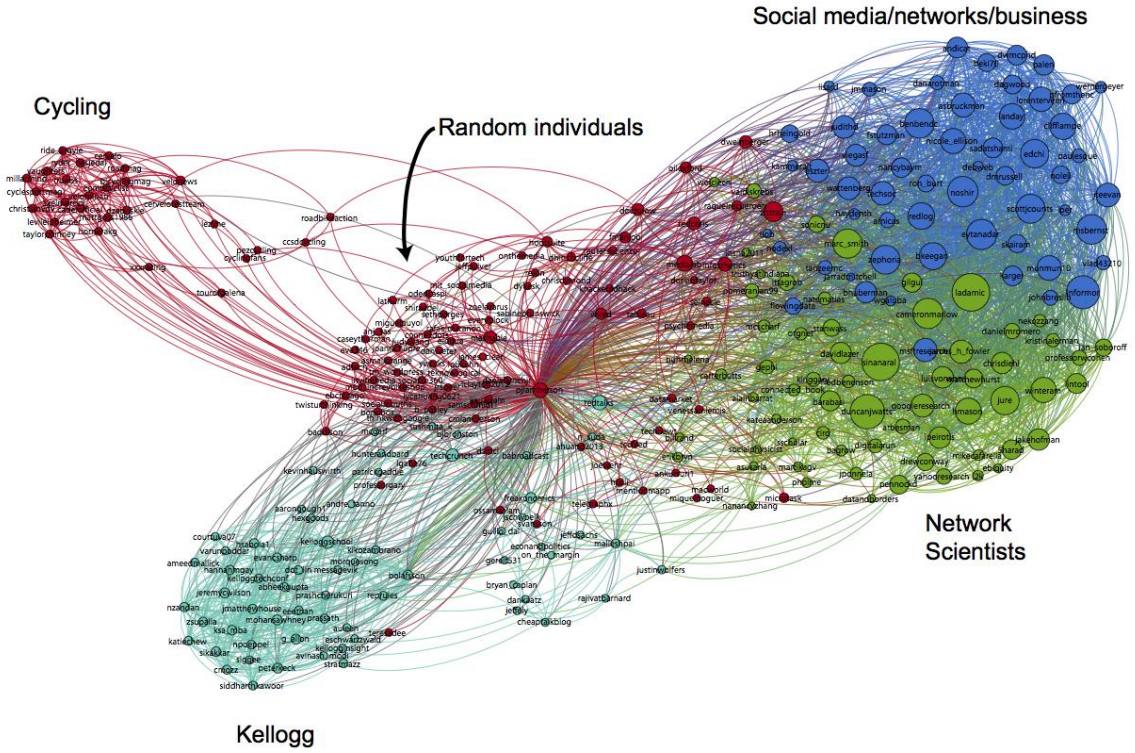

Figure 7. Twitter Network Visualization (Gephi, 2014).

2.7.2 D3 JavaScript Visualization

Data-Driven Documents (D3) is a JavaScript library designed to manipulate interactive data through Hyper-Text Markup Language (HTML), Cascading Style Sheets (CSS), and Scalable Vector Graphics (SVG) inside a web browser. Data is bound to a Document Object Model (DOM) and transformed to display various visualizations. The D3 library is especially appropriate for visualizing digital forensic data considering its ability to support large datasets with a relatively small overhead (Bostock, 2014).

Figure 8 shown below contains an example of a Bubble Chart visualization. This particular image depicts the frequency of words contained in the D3 JavaScript website homepage (www.d3js.org). Larger bubbles represent words that occur more often than others.

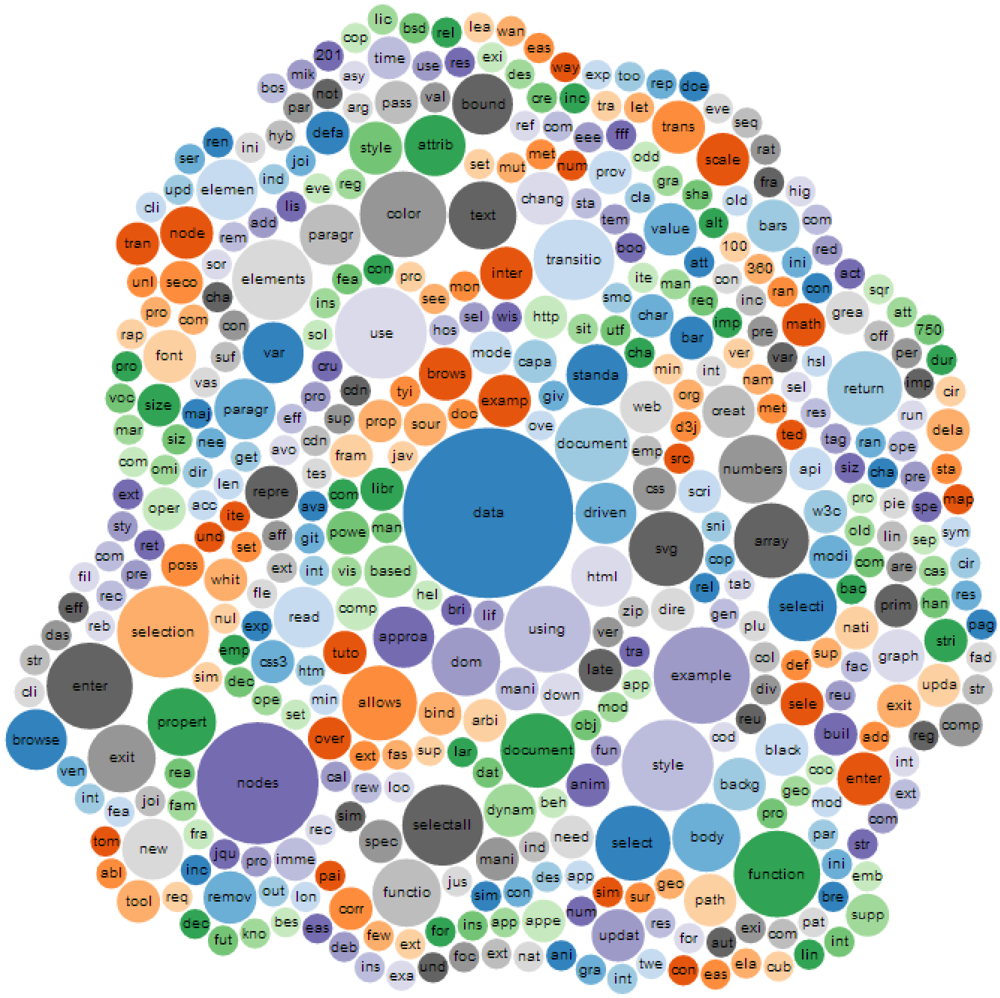

Figure 8. Bubble Chart of D3 Homepage (Bostock, 2014).

Figure 9 illustrates the political contributions for the 2012 Election Cycle using colored bubble nodes to represent the candidates and arc slices to represent each Political Action Committee (PAC). Bubble and path sizes are proportional to contribution amounts from

PACs to political candidates. Interaction with the visualization allows the user to explore datasets from both the House of Representatives and the U.S. Senate.

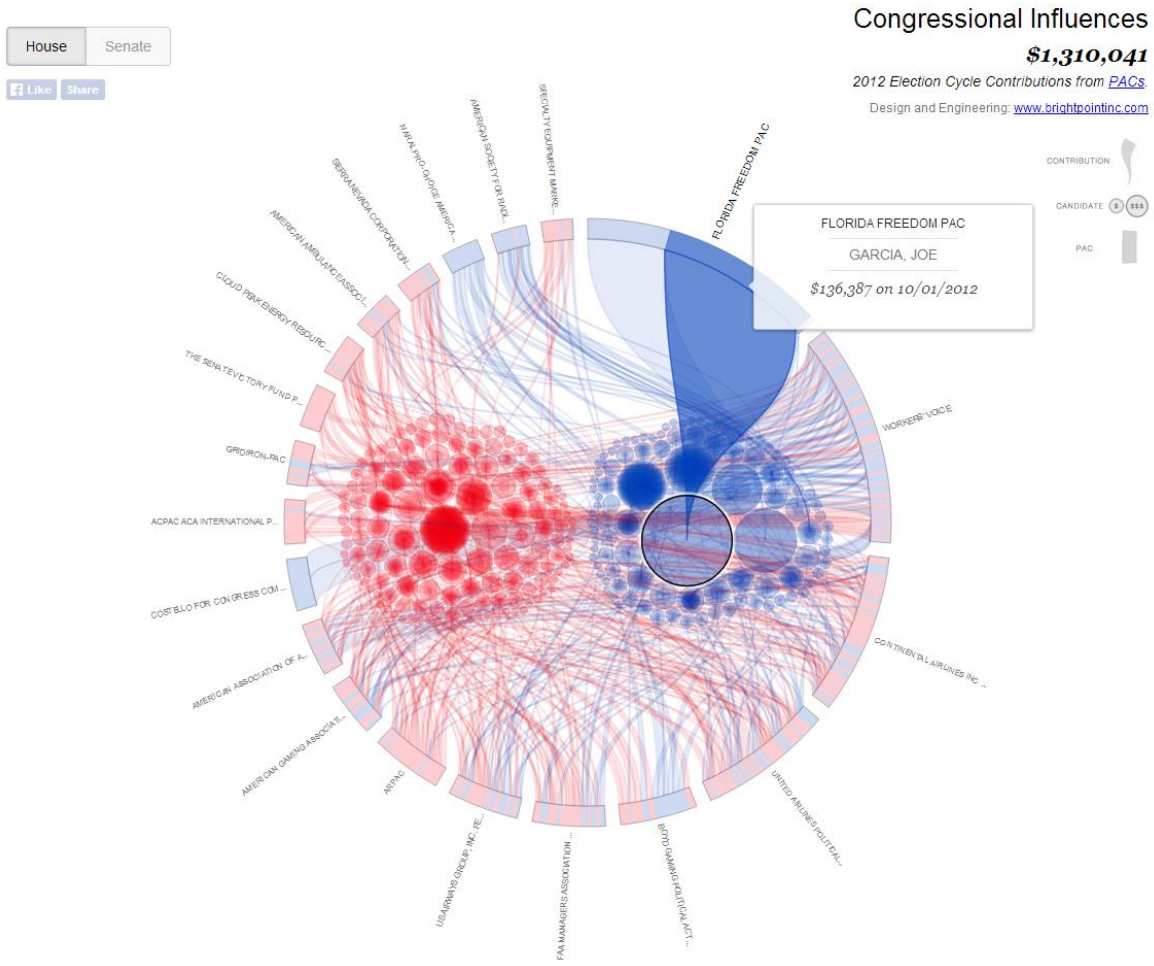

Figure 9. Political Contributions 2012 Visualization (Brightpoint, 2014).

Figure 10 provides a static image from another interactive visualization which details historical data from the National Association of Securities Dealers Automated Quotations (NASDAQ) stock exchange. This example demonstrates the ability of the D3 JavaScript

Library to portray multiple visualizations including histograms, bar charts, pie charts, and circle graphs in one interactive view.

For the purposes of this research, the D3 JavaScript library was selected to create interactive visualization. The main advantage to utilizing D3 is its ability to run in most web browsers making the tool developed in Chapter 3 extremely portable.

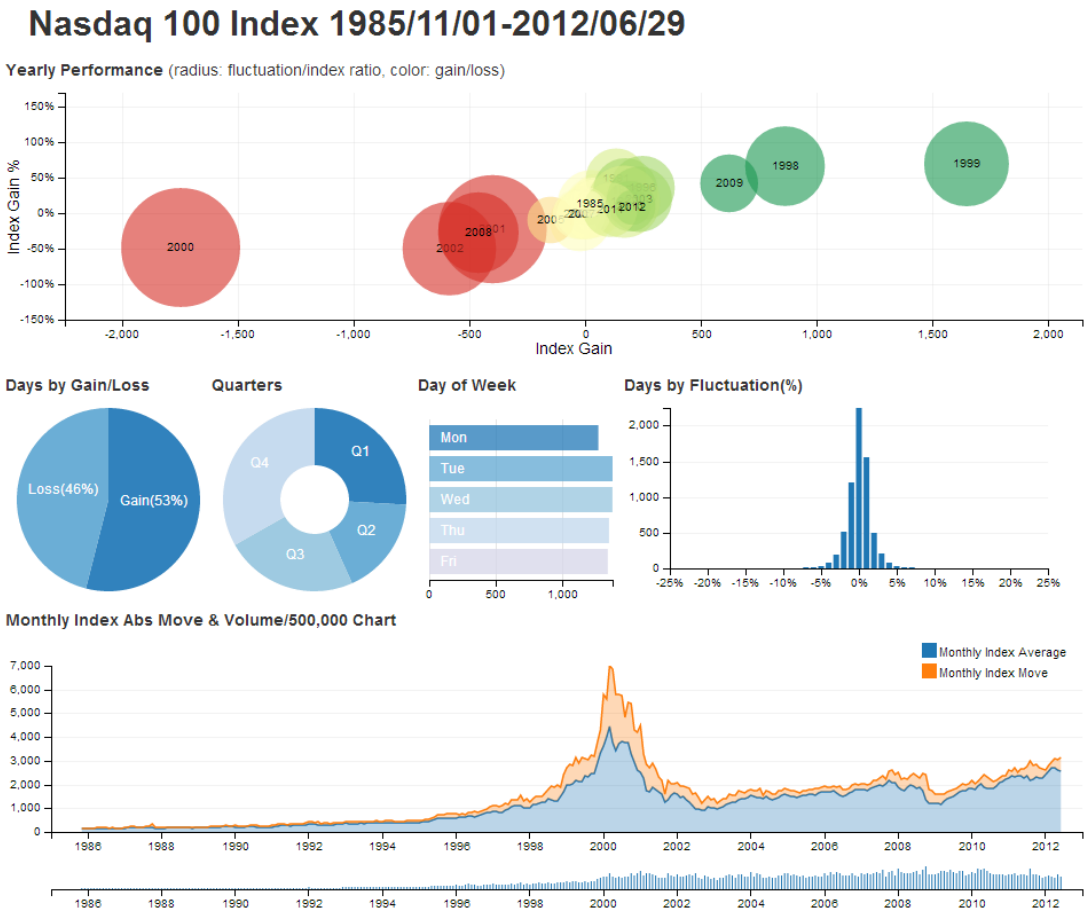

Figure 10. NASDAQ Interactive Chart (NASDAQ, 2014).

2.8 Summary

This chapter described the process of digital forensics with an emphasis on the specific field of Forensic Memory Analysis. Topics discussed include the significance of

memory analysis, methods of acquiring RAM dumps from Windows computers, and open source memory artifact extraction tools. The applied science of Information Visualization is explored as a possible solution for overcoming some of the challenges associated with large and complex datasets. Both the Gephi software package and the D3 JavaScript library are examined as potential interactive visualization tools. D3 is ultimately selected for use in this research due to its versatility and portability.

III. Methodology

Various open-source programs exists which are capable of extracting digital evidence from forensic memory dumps. While many of these tools provide textual output to the user, locating key information in the digital evidence proves challenging. With the increasing complexity of digital forensic collections, analysts find it difficult to understand the unique state of the system at the time of memory capture. This research applies Information Visualization techniques to the forensic memory analysis process by providing analysts with both global and local views of the extracted data. The interactive tool assists in identifying items of interest and recognizing behavior patterns in the data.

This chapter describes the specific goals of the research as well as the design approach to achieving the outlined objectives. The process for obtaining forensic memory captures from Windows systems is discussed and the method of extracting digital artifacts is outlined. Additionally, the chapter provides the Information Visualization tools and techniques used to develop an interactive tool for forensic memory analysis. The basic functionality of the tool is covered in along with the specific features that provide unique visual support to the user.

3.1 Research Objectives

The goal of this research is the development of a proof-of-concept model for visualizing data extracted from a forensic memory image. The specific objectives are outlined as follows:

1. Represent multiple datasets extracted from a forensic memory image in an interactive visual with one universal view of the target system.

2. Filter a global dataset for a particular system and visually express the following associations:
    - The relationship between a specific system process and its associated resources.
    - The relationship between a specific system resource and its associated processes.
3. Assist in the visual identification of unique patterns and new pieces of information from datasets.

3.2 Assumptions

Assumptions related to achieving the objectives include:

1. Digital forensic methods exist for obtaining an accurate physical memory image.
2. The extracted datasets under consideration for each system are limited to process lists, network connections, system services, open file handles, system registry keys, and loaded modules.
3. The operating systems under consideration is limited to Microsoft Windows XP.

3.3 System Design

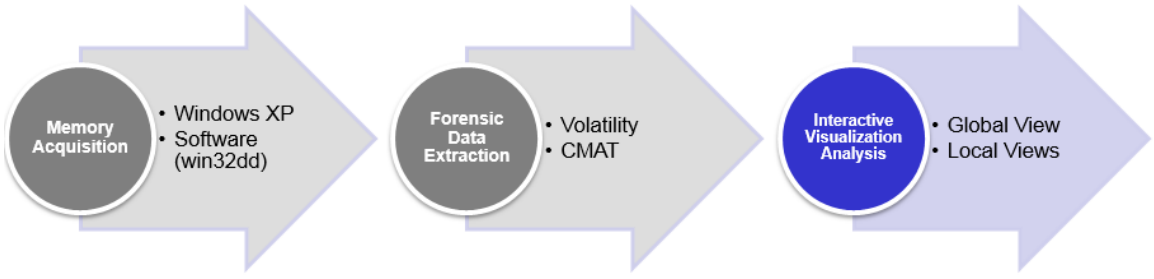

Figure 11. Forensic Memory Analysis Process.

Figure 11 illustrates the three step process for collecting, extracting, and analyzing forensic memory images. The primary focus of this research is the third step of the process which involves the development of a software tool capable of generating interactive visualizations from datasets extracted from Windows RAM dumps. The overall objective is to provide a functional system that is flexible enough to support future datasets from various operating system platforms and architectures. Additionally, the interface design should be simplistic and easy to use for both skilled and non-skilled Digital Forensic Analysts. The visualization tool should have the ability to operate on a range of workstation platforms used for forensic analysis.

3.3.1 Memory Acquisition

Several Windows-based computers provide the source memory data for testing the visualization tool. Each machine is created and configured to produce unique process activity. The test computers are VMWare virtual machines. Table 4 contains a summary of the test machines and their configurations. Forensic memory dump acquisition is performed via a software-based approach by running `win32dd` on each machine.

3.3.2 Malware Background

This section provides specific background information regarding the two types of malware used in this research. The malware include FUTo and Poison Ivy, a Remote Administration Tool (RAT).

Rootkits maintain access to a system by hiding their activity from the system user. The FUTo rootkit, for example, removes references to itself from the active process list data structures (Stevenson & Altholz, 2006). Unlike many rootkits that employ forms of

Table 4. Test Machine Configurations.

Dataset Label	Operating System	Unique Process Activity
WinXP IE	Windows XP SP3	Active Internet Explorer (v. 8.0.6001.18702) Process
WinXP Chrome	Windows XP SP3	Active Google Chrome (v. 34.0.1847.131) Process
WinXP Firefox	Windows XP SP3	Active Mozilla Firefox (v. 28.0) Process
WinXP Solitaire	Windows XP SP3	Active Solitaire Process
WinXP Word	Windows XP SP3	Active Microsoft Word Process
WinXP Malware 1	Windows XP SP3	FUTo Malware hiding Solitaire Process
WinXP Malware 2	Windows XP SP3	Remote Administration Tool Malware (hidden)
WinXP Malware 3	Windows XP SP3	Remote Administration Tool Malware (hidden using different process)

API-hooking, FUTo uses Direct Kernel Object Manipulation (DKOM) to directly modify kernel structures including the PspCidTable (Silberman & C.H.A.O.S., 2005).

Poison Ivy is a Remote Administration Tool piece of malware that can perform malicious activity such as “key logging, screen capturing, video capturing, file transfers, password theft, system administration, traffic relaying, and more” (Bennett, et al, 2013). Poison Ivy can also be configured to inject itself into a browser process. This allows the RAT to effectively bypass the firewall and make outgoing connections at will.

3.3.3 Forensic Data Extraction

Each forensic memory image is processed by the Volatility Framework (discussed in Section 2.5.1) to extract digital artifacts. Table 5 lists the specific plugin modules executed on each memory dump file. The tool exports each dataset as a text file which is labeled according to content and source machine (i.e., Process_List_Chrome.txt, Connections_Firefox.txt).

The images are also processed using the CMAT utility (see Section 2.6.2) to extract data and generate the associated feature files. These feature files are compared against the Volatility output files for discrepancies in the data. Unfortunately both CMAT and Volatility lack the capability to extract data from a memory image relating to Running Services. For the purposes of this research, the information is obtained via the `tasklist` command-line tool included in Windows installations. The tool is executed at the time of memory acquisition in order to obtain a complete dataset. Sample output is depicted in Figure 12 below. Each listing is output to a text data file.

Table 5. Volatility Plugins Executed on Test Images.

Module Plugin	Description
connections	View the TCP connections active at the time of memory acquisition
handles	Display the list of open handles for all processes
pslist	List the active processes of the system
sockets	View the listening network sockets for any protocol

```
C:\Users\Agent>tasklist /svc

Image Name                     PID Services
========================= ======== ================
System Idle Process              0 N/A
System                           4 N/A
smss.exe                       372 N/A
csrss.exe                      504 N/A
wininit.exe                    580 N/A
csrss.exe                      604 N/A
services.exe                   644 N/A
lsass.exe                      664 KeyIso, Netlogon, SamSs
lsm.exe                        696 N/A
winlogon.exe                   744 N/A
svchost.exe                    828 DcomLaunch, PlugPlay, Power
nvvsvc.exe                     892 nvsvc
nvSCPAPISvr.exe                916 Stereo Service
svchost.exe                    960 RpcEptMapper, RpcSs
MsMpEng.exe                    160 MsMpSvc
svchost.exe                    592 AudioSrv, Dhcp
```

Figure 12. `tasklist` Command-Line Output.

3.3.4 Data Formatting

The extracted datasets for each Memory Dump require conversion from text files into the JSON format that can be used by the D3 JavaScript visualization software. This is accomplished by importing each text file into a Microsoft Excel spreadsheet and subsequently exporting as a comma separated values (CSV) file. Because this research is a proof-of-concept, the design lacks an automated process for completing this task and must be performed manually. The correctly formatted CSV output files are directly imported into the Visualization tool described in the next section.

3.3.5 Visualization Tool Design

The tool developed in this thesis utilizes the D3 JavaScript visualization library to manipulate HTML, CSS, and SVG objects within a web environment. The advantage of utilizing JavaScript in an active web environment is the ability to use the visualization

tool on most web browsers. For testing purposes the tool is hosted locally through a simple Apache web server.

Table 6 contains a listing of the source files included in the visualization tool working directory along with brief descriptions of each resource. The main effort of this research involved writing and debugging the source files which contain approximately four thousand lines of JavaScript code. The only source file not authored in this research is the D3 JavaScript Library file. A sequence diagram outlining basic functionality of the developed visualization tool is depicted in Figure 13.

Table 6. Visualization Tool Source Files.

File Name	Description
index.html	HTML source file
globals.js	Global variable definitions
buttons.js	Interactive button action definitions
initialize.js	Functions for initializing the interface
update.js	Functions for updating the visualization
get.js	Data accessor functions
events.js	Mouse click event functions
d3.v3.min.js	D3 JavaScript Library
style.css	File specifying CSS Style options

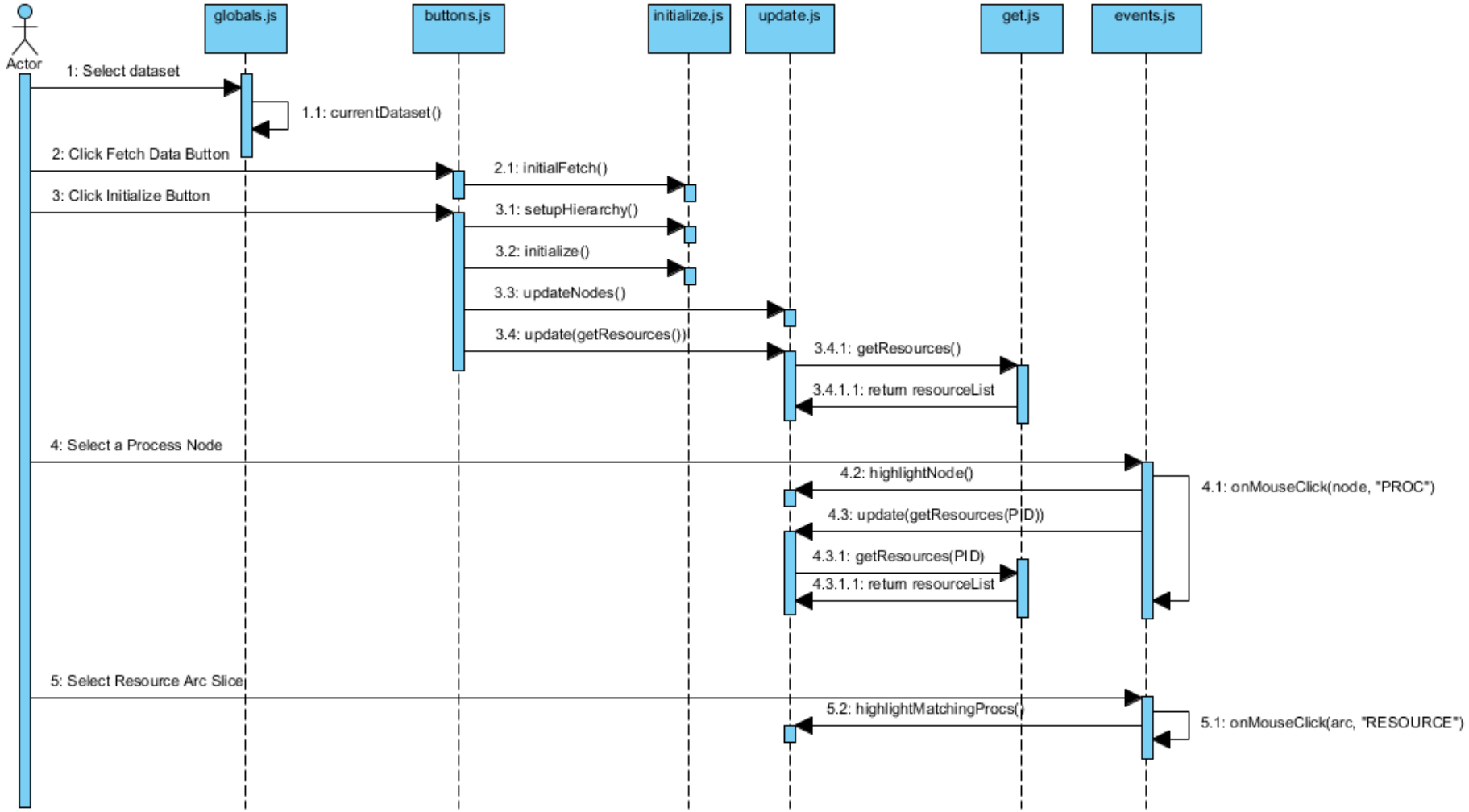

Figure 13. Visualization Tool Sequence Diagram.

3.3.6 Visualization Tool Layout

Figure 14 provides a view of the main user-interface with outlines highlighting the three distinct regions of the interface. The top of the page contains a dataset selector menu and multiple buttons. The center main region contains the interactive visualization while the green pane on the right displays instruction and selected data details.

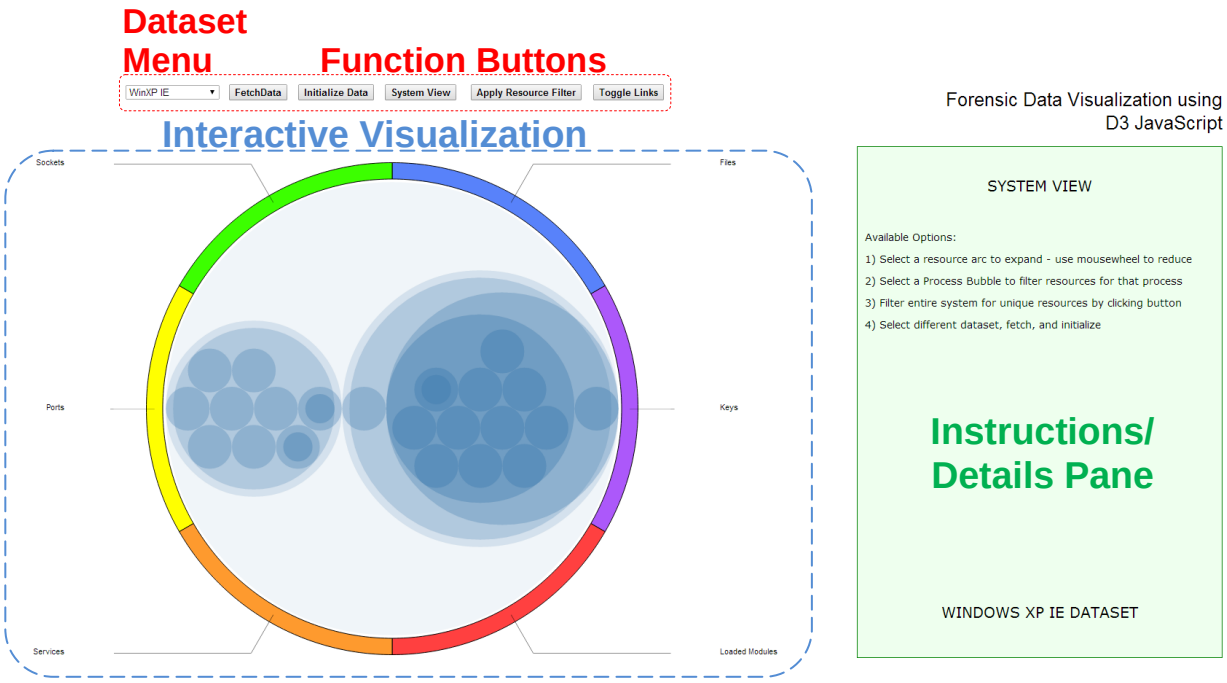

Figure 14. Visualization Interface Layout.

The interactive visualization region is divided into three major components (Figure 15):

- Process Nodes – Hierarchical layout containing bubble nodes representing system processes.
- Resource Arcs – System resource lists divided into arcs and color coded according to type which include File Handle, Key Handles, Loaded Modules, Services, Ports, and Sockets.

- Links – Colored links connecting resources and their associated process nodes.

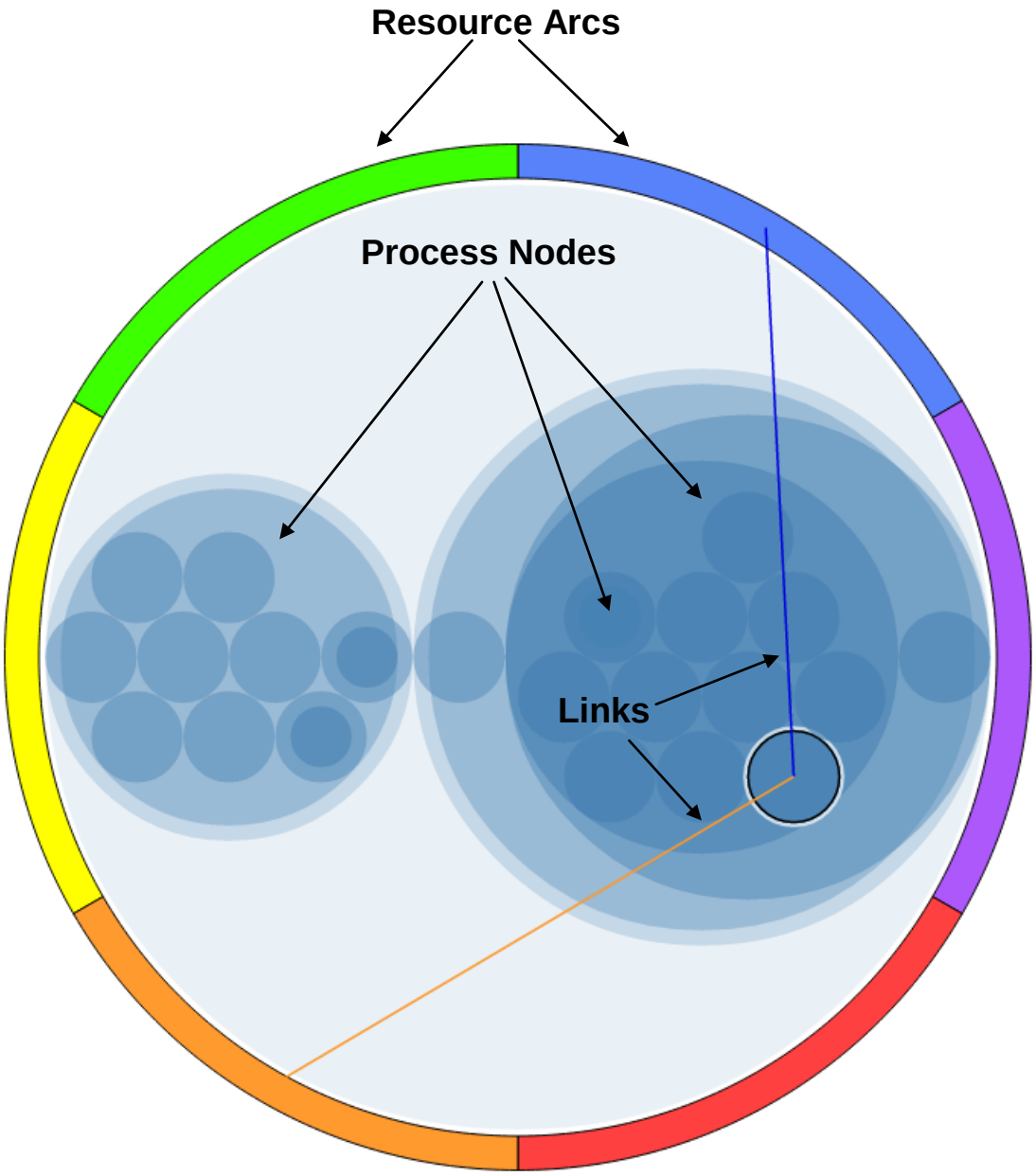

Figure 15. Visualization Region Components.

3.4 Visualization Tool Functionality

This section provides steps detailing the basic functions of the visualization tool.

3.4.1 Dataset Selection and Initialization

The following steps are used to select and initialize datasets:

1. Click the drop-down menu and select a dataset.
2. Click the Fetch Data button to import the appropriate CSV data files.
3. Click Initialize Data button to initiate the interactive visualization.

3.4.2 Visualization Interaction

The visualization tool is designed to follow a process flow similar to the Explore, Investigate, and Correlate (EIP) Process described in Chapter 2. Upon initialization, the visualization depicts a high-level *global view* of the system. The resource arcs are populated with a complete listing of system resources which the user can Explore by clicking to expand as shown in Figure 16.

Interacting with the visualization allows the user to Investigate areas of interest by means of process node highlighting and resource arc slice filtering. The resulting *local view* provides a low-level perspective of the system by displaying the connections between processes and resources. By investigating areas of interest, the user discovers highlighted relationships within the datasets. This can be accomplished in three ways:

1. Clicking on a Process Bubble Node (see Figure 17)
    - Process Node is highlighted.
    - Resource arcs are filtered to list files, keys, modules, services, ports, and sockets associated with the selected process.

2. Clicking on a Resource Slice (see Figure 18)

    - Process Nodes accessing selected resources are highlighted.

3. Clicking the Apply Resource Filter button (see Figure 19)

    - Resource arcs are filtered to contain all open ports, open sockets, user files, and executable loaded modules for all system processes.

The final feature of the visualization allows users to both highlight specific relationships as well as identify behavior patterns by overlaying colored links from an expanded list of resource slices to all related process nodes. This is accomplished by selecting the Toggle Links button at the top of the interface while in an expanded resource list view (i.e., currently viewing expanded Sockets List) as shown in Figure 20. This mechanism assists the user in discovering Correlations within the data through visual aids.

Additional features of the visualization include:

- Tooltips displaying process names on mouse over
- Tooltips displaying full resource labels on mouse over
- Process summary data displayed in green data details pane when node selected (see Figure 17)

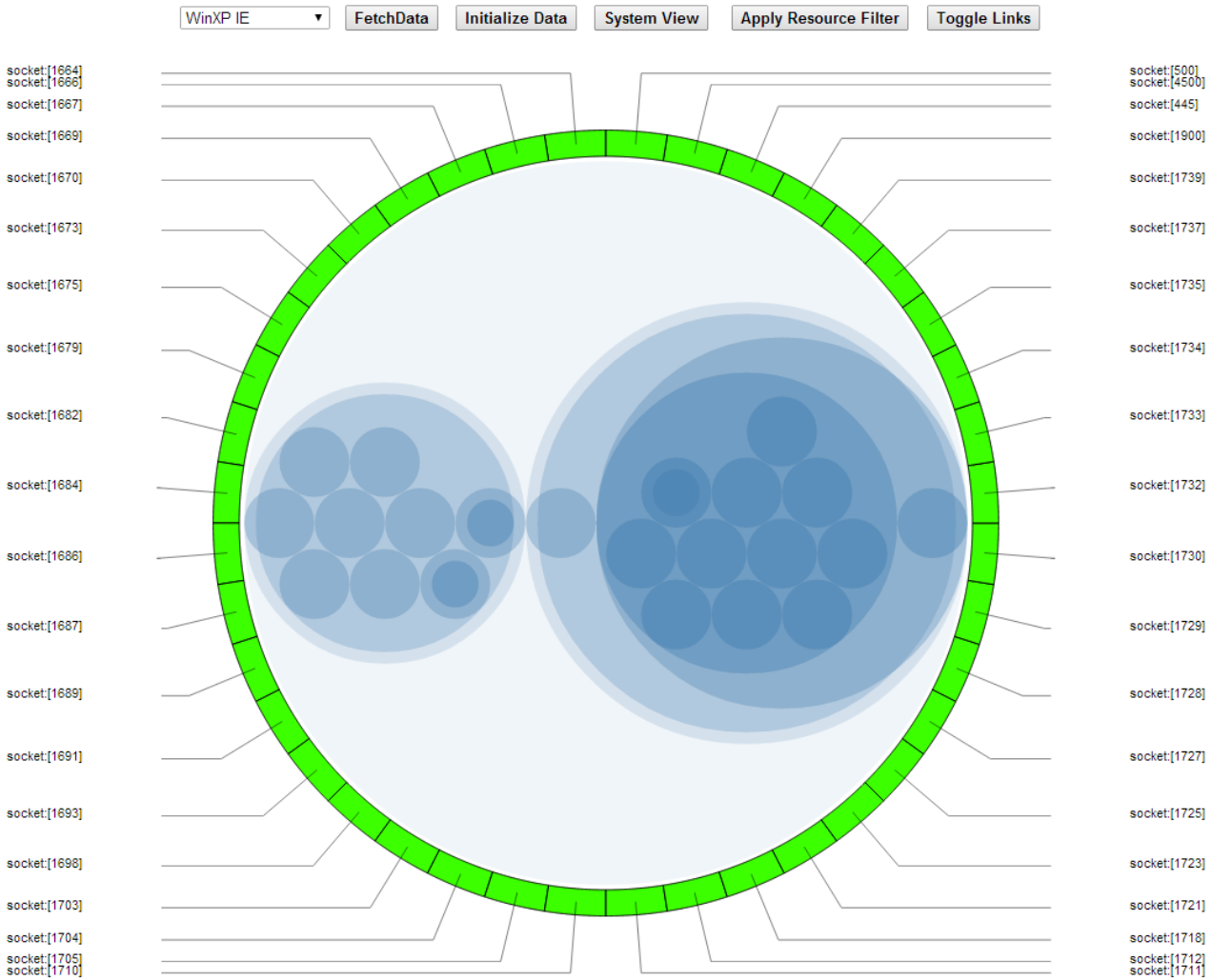

Figure 16. Expanded Resource Arc – Sockets List.

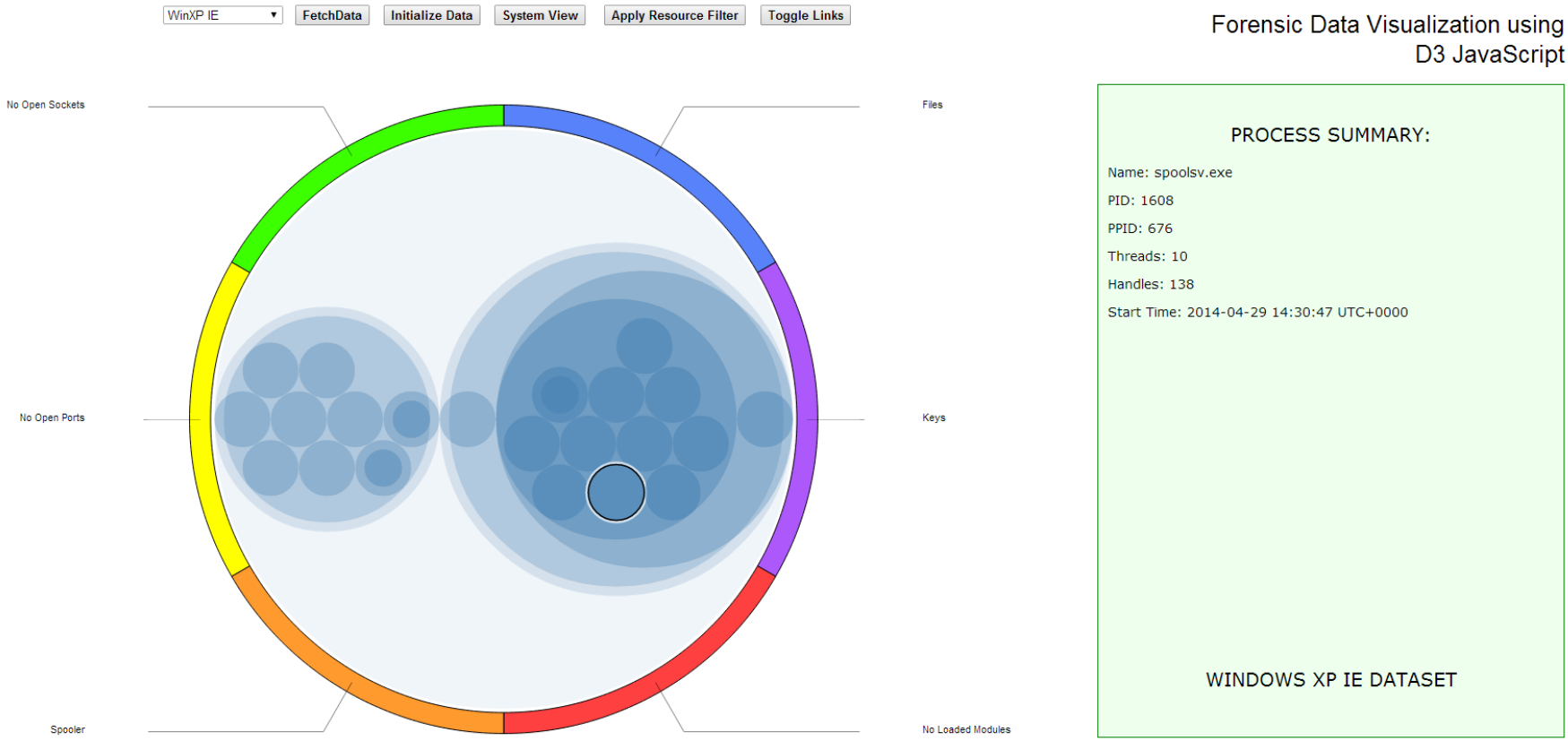

Figure 17. Process Bubble Selected – Resource Arcs Filtered.

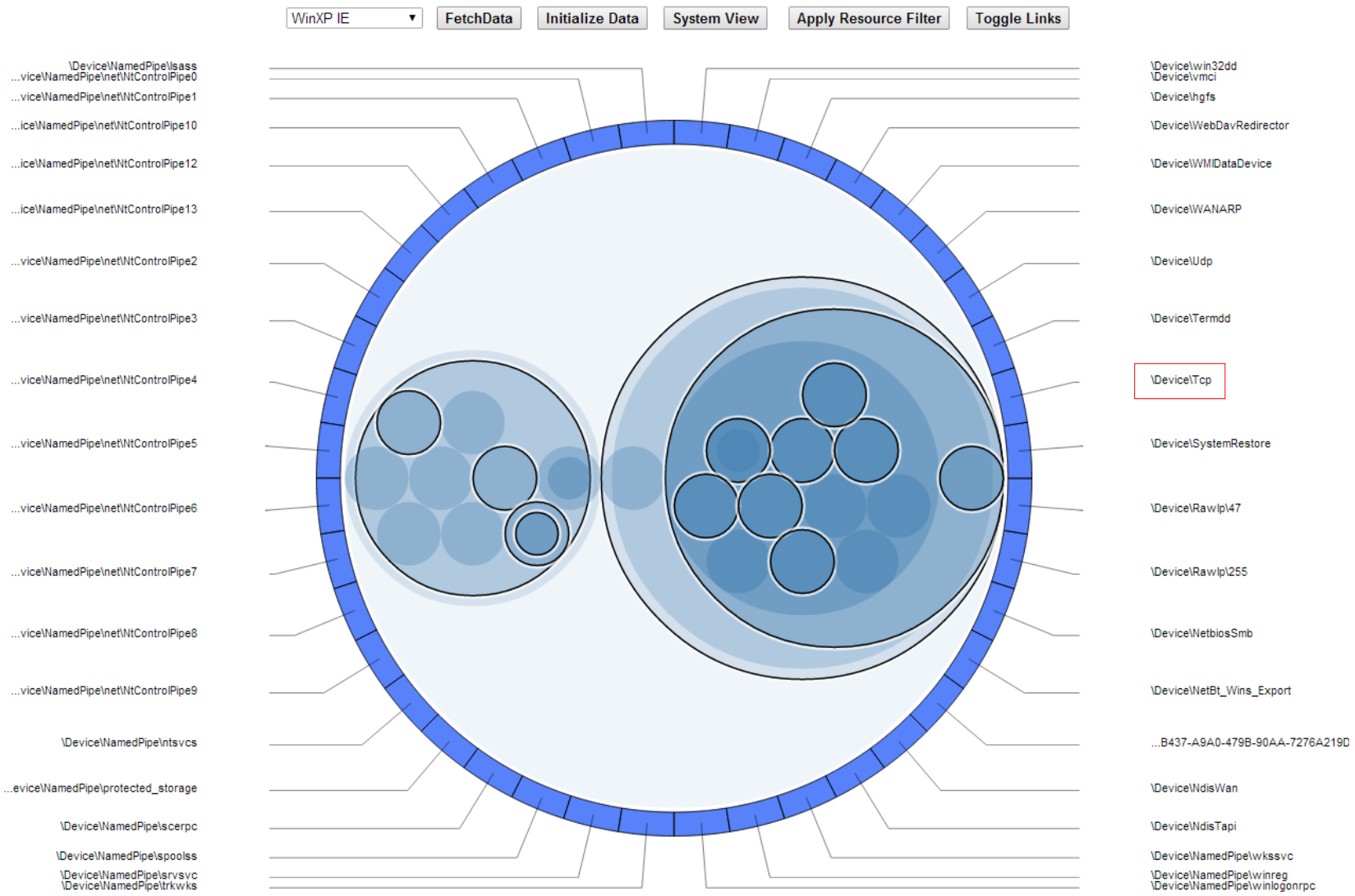

Figure 18. Resource Slice Selected – Associated Process Nodes Highlighted.

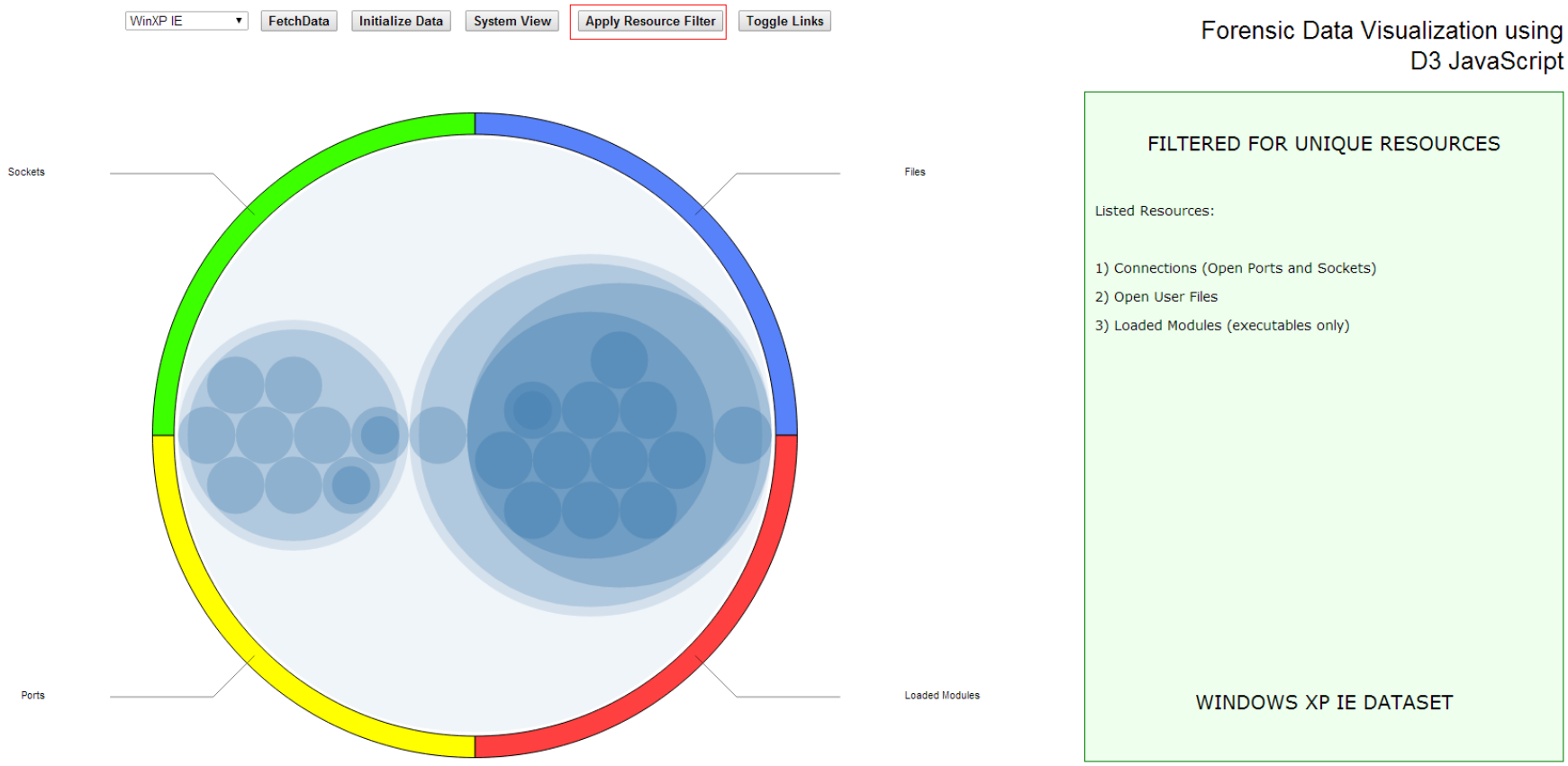

Figure 19. Apply Resource Filter Button Selected.

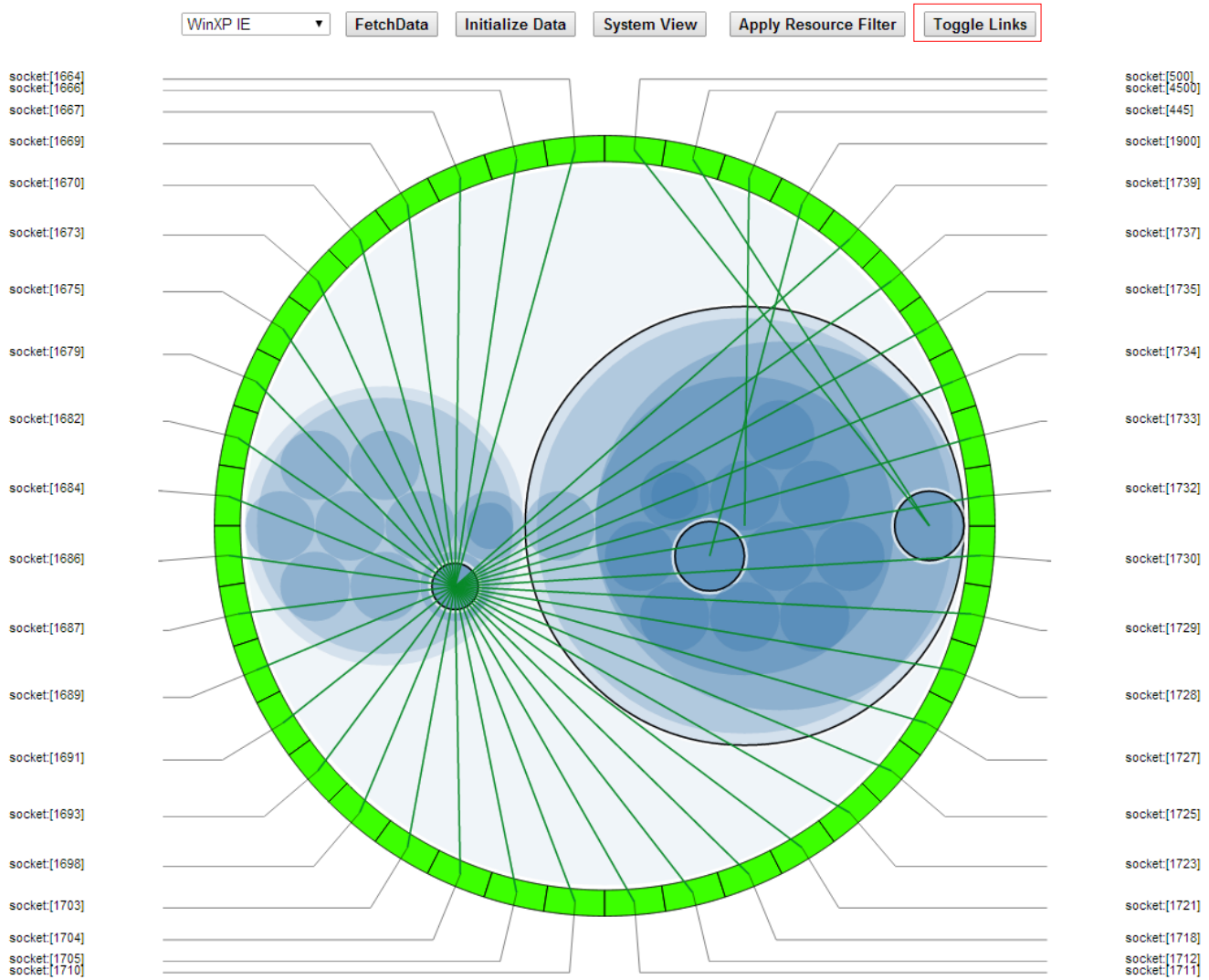

Figure 20. Links displayed – Sockets List Expanded.

3.5 Summary

This chapter outlined the research objectives and specific design approach to accomplishing those goals. The process for obtaining forensic memory captures from Windows machines is discussed along with the techniques used to extract digital artifacts from the dumps. The two types of malware that exist in the test memory dumps are reviewed to outline their basic behavioral characteristics.

The last portion of the chapter provides a functional overview of the Visualization tool user interface. Basic features of the software are presented including the global system view and potential user interactions resulting in a more local view of the system. The addition of visual links to connect resources and associated processes assists the user in locating unique patterns and information of significance within the digital forensic datasets.

IV. Analysis and Results

In order to assess the abilities of the visualization tool developed in this research, test forensic memory images are created and acquired. The forensic memory analysis tools previously described are then executed to extract digital artifacts from the dumps. The resulting datasets are formatted using Microsoft Excel for import into the visualization tool.

This chapter describes the application of the D3 JavaScript Visualization tool to various test datasets acquired from a number of Windows XP memory dumps. The results from each of the sample datasets are evaluated against the three research goals. The first goal involves providing a global view of the data that is extracted from forensic memory dumps using memory analysis tools. The second goal requires that the visualization tool possess the ability to filter the forensic datasets and highlight two types of relationships. These include the association between a process and its corresponding resources, and the connections between a particular resource and the processes accessing it. The third goal states that the visualization tool should contain features that assist the user in identifying unique patterns and interesting items within the datasets. Each objective is discussed in detail in the following sections.

4.1 Goal 1: Global View of Data from Memory Analysis Tools

The first goal requires the visualization tool provide an interactive global view of the data output from forensic memory image analysis tools. Ultimately this visualization should allow the user to explore the entire set of available information extracted from a memory capture.

The visualization tool meets this objective by:

- Providing a global view of the target system based on the datasets extracted from the respective memory dump.
- Populating the resource arcs with lists of complete system resources based on six categories (files, registry keys, loaded modules, services, ports, and sockets).
- Allowing the user to explore the resources by expanding/reducing each resource arc slice.

This goal is met by the full process hierarchy (node layout) and global resource listing (expandable arcs) as shown in Figure 21. For this particular dataset there are a total of 34 processes, 374 unique file handles, 268 registry key handles, 349 loaded modules, 44 active services, 3 open ports, and 15 open sockets which populate the visualization.

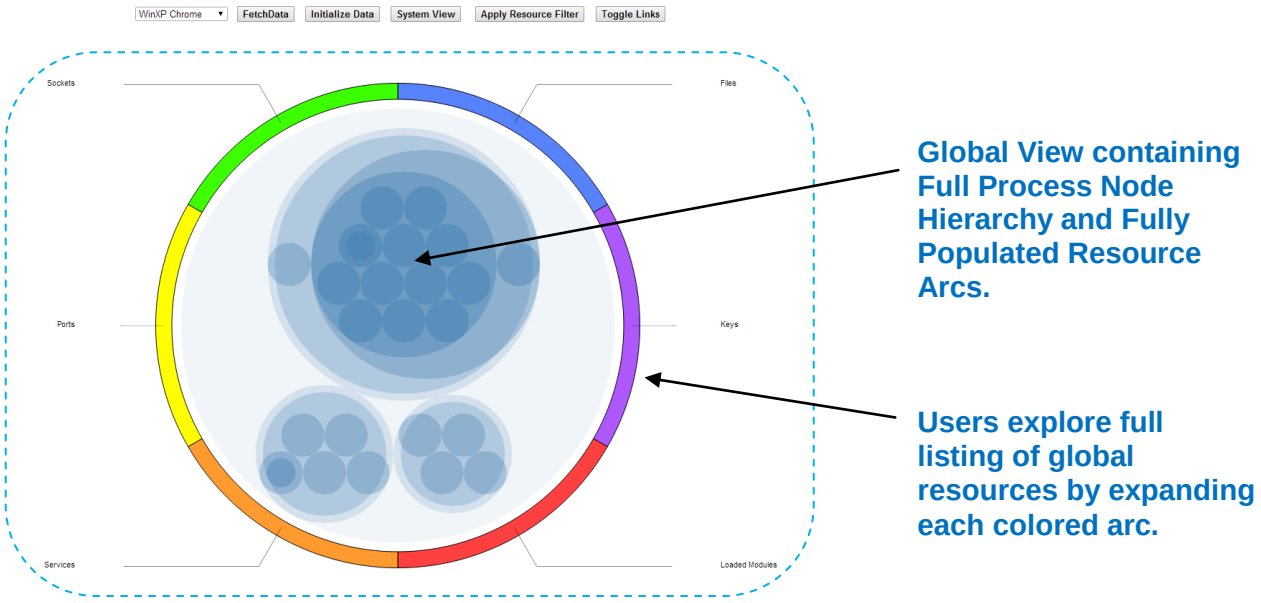

Figure 21. Overall System View.

4.2 Goal 2: Filter Data and Display Relationships

The second goal states that the visualization tool should possess the ability to filter the datasets and highlight the relationship between:

- A single process and its associated resources (Figure 22)
- A single resource and its associated processes (Figure 23)

These goals are evidently satisfied as portrayed by the selected process and associated resource arcs (Figure 22) as well as the selected loaded module resource and associated processes (Figure 23).

The purpose of this objective is to understand the behavior of the system by visualizing the low-level connections between objects. It is through this local view of the data that relevant information is discovered.

4.3 Goal 3: Assist in Identifying New Data and Patterns

The third research goal states that the visualization tool should assist the user in identifying unique patterns and new pieces of information within the datasets. To verify this objective, each dataset contains unique process activity at the time of memory capture (see Table 4). There are two methods that can be applied to spot unique activity or new data:

1. Toggle links to display multiple resource relationships (limited to single resource type) in one view.
2. Compare trusted system visualization (containing only legitimate process activity) to an untrusted system visualization (containing a hidden process, rootkit, or other malware).

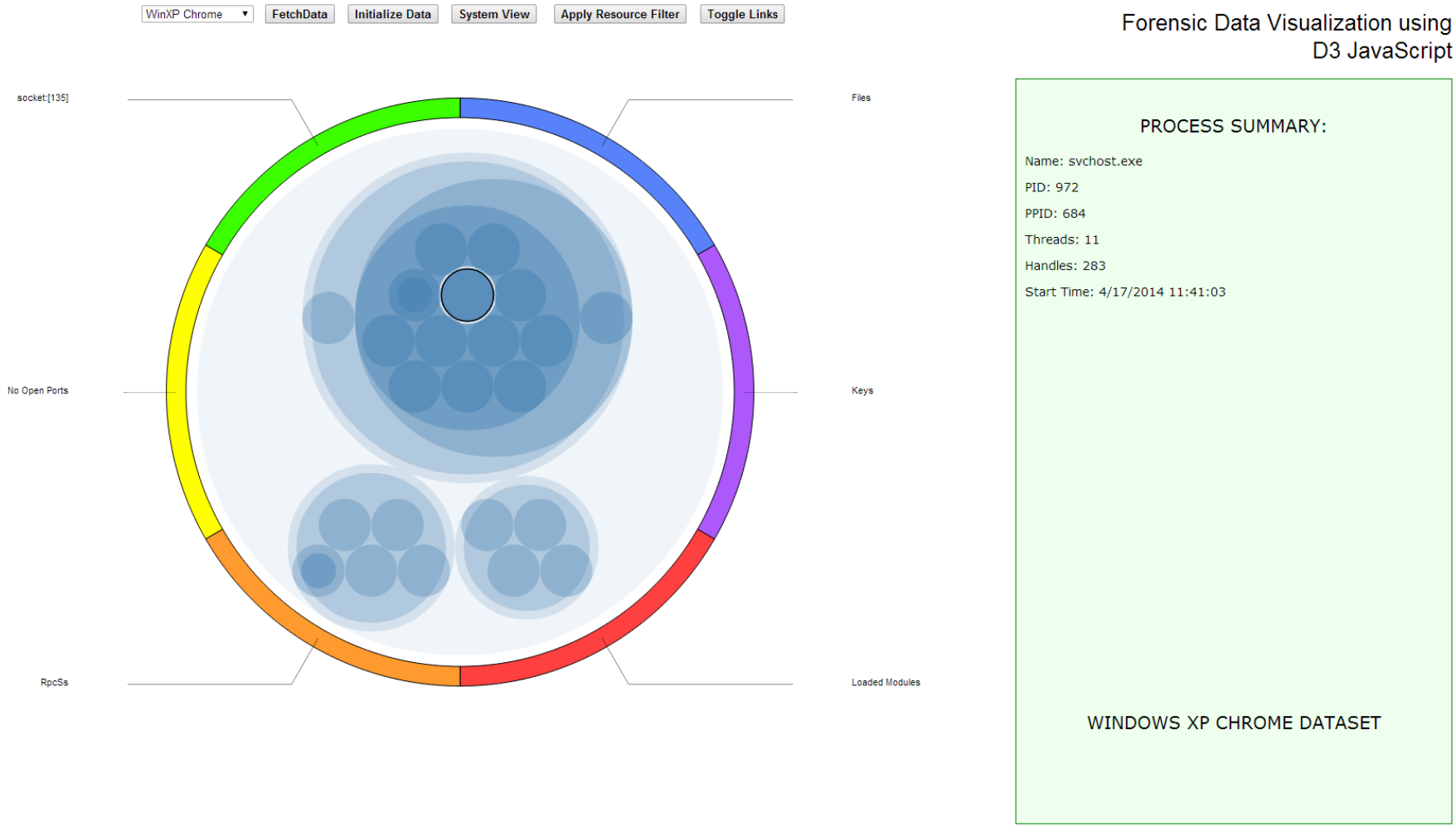

Figure 22. Visualization filtered for single process.

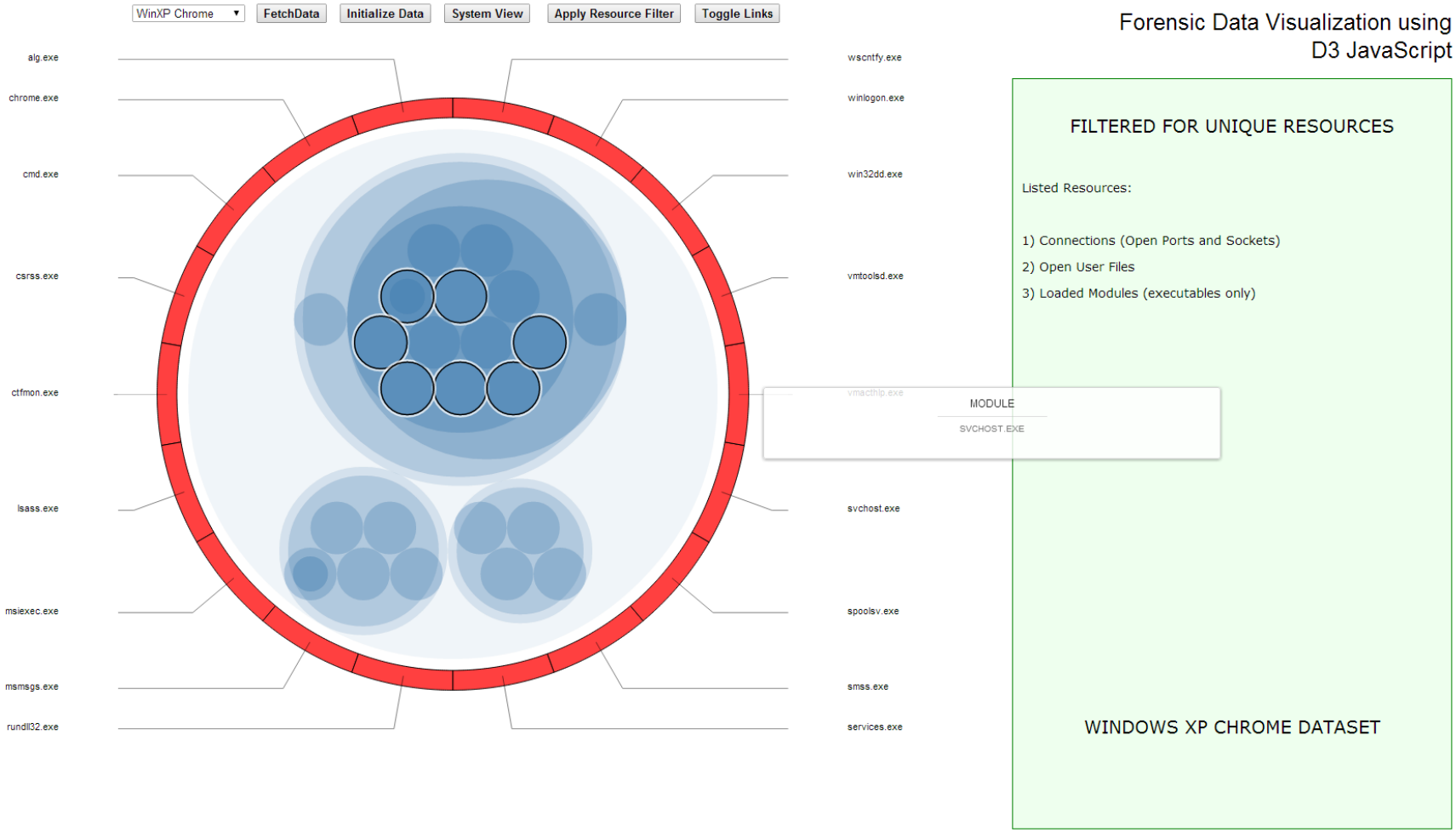

Figure 23. Visualization filtered for single resource.

Four datasets are labeled as trusted systems running a single instance of a legitimate process while three other datasets contain some form of malware. The above methods are applied to evaluate the third overall research goal.

4.3.1 Windows XP SP3 (Internet Explorer) – Trusted Image

The first memory image consists of a single instance of Internet Explorer running on a Windows XP SP3 machine. Applying the unique resource filter provides a list of connections (ports and sockets), user files, and loaded modules. Clicking on the Socket and Port Resource arcs populates the resource ring with the respective resource lists. Using method 1 from above, toggling the links when viewing both lists produce visualizations that provides the user with some unique patterns (see Figures 24, 25). There are numerous connections surrounding one particular process, which upon examination is the `IEXPLORER.EXE` process running inside another instance of `IEXPLORER.EXE`.

Filtering the global resource list by clicking the *Apply Resource Filter* button and selecting the red loaded modules arc slice populates the resource ring with the list of executable loaded modules. Toggling the links for this list provides another view (see Figure 26). By browsing this short list the user can identify the `iexplorer.exe` module and verify that it is loaded by the appropriate `IEXPLORE.EXE` process node.

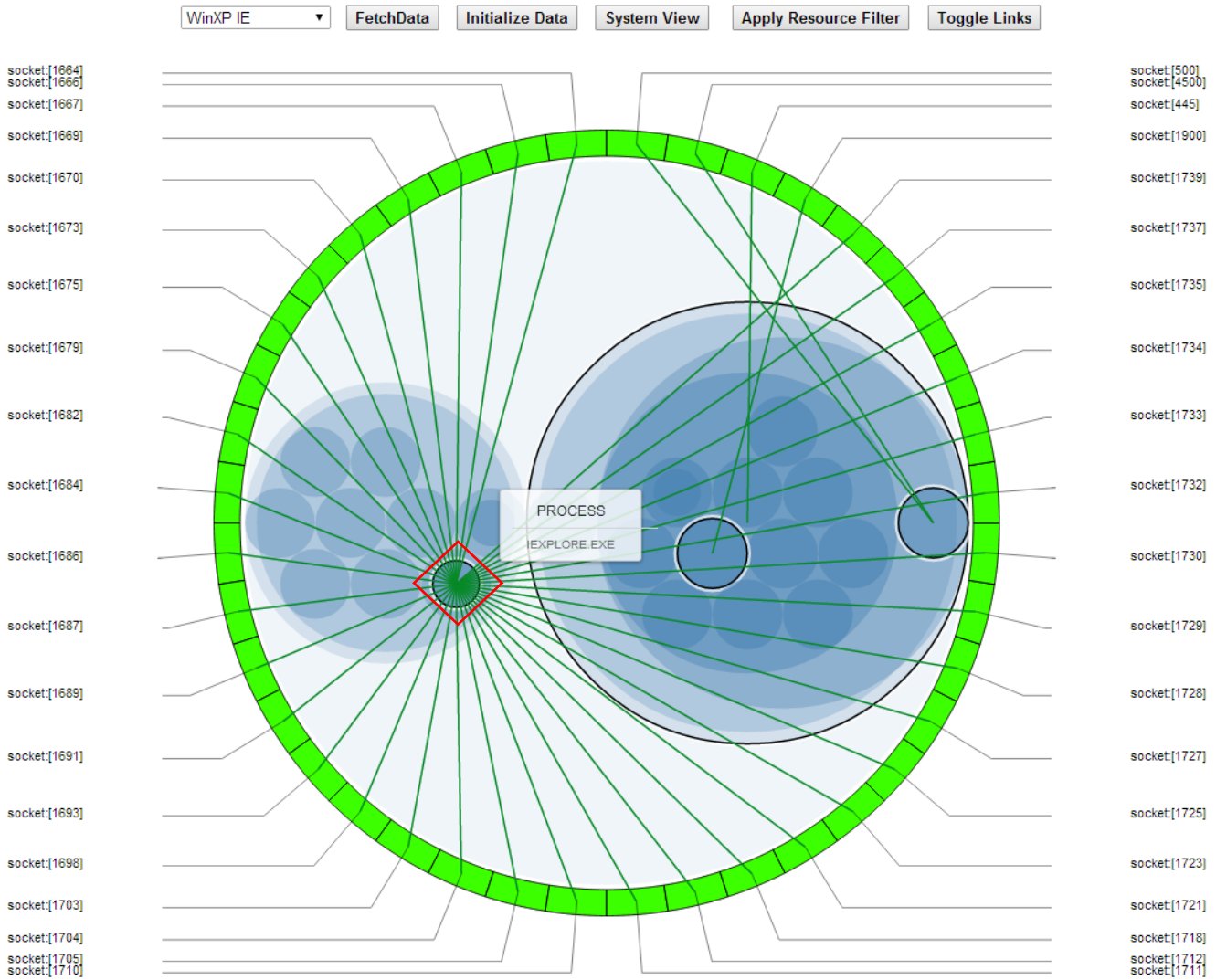

Figure 24. IE – Socket List Links.

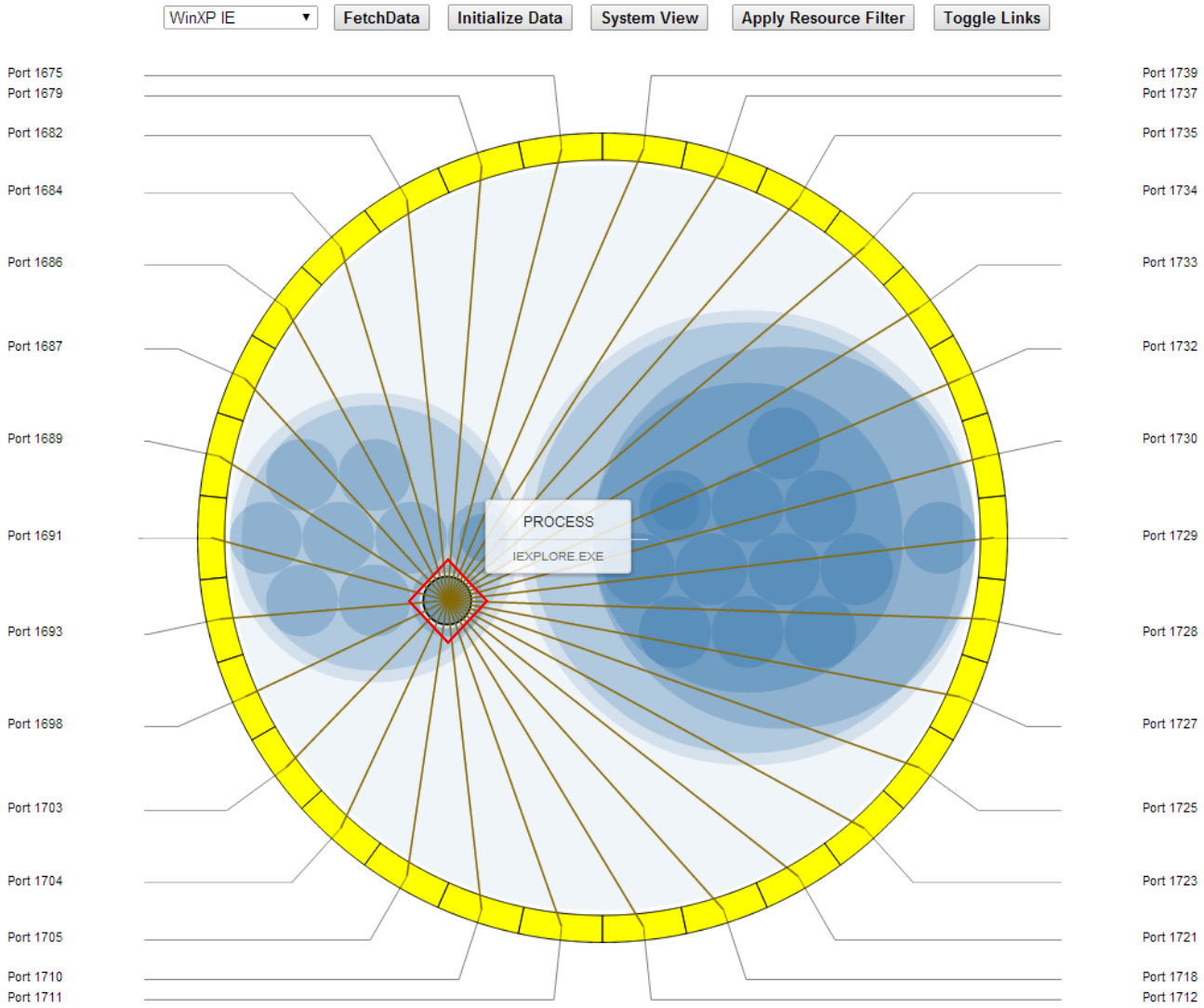

Figure 25. IE – Port List Links.

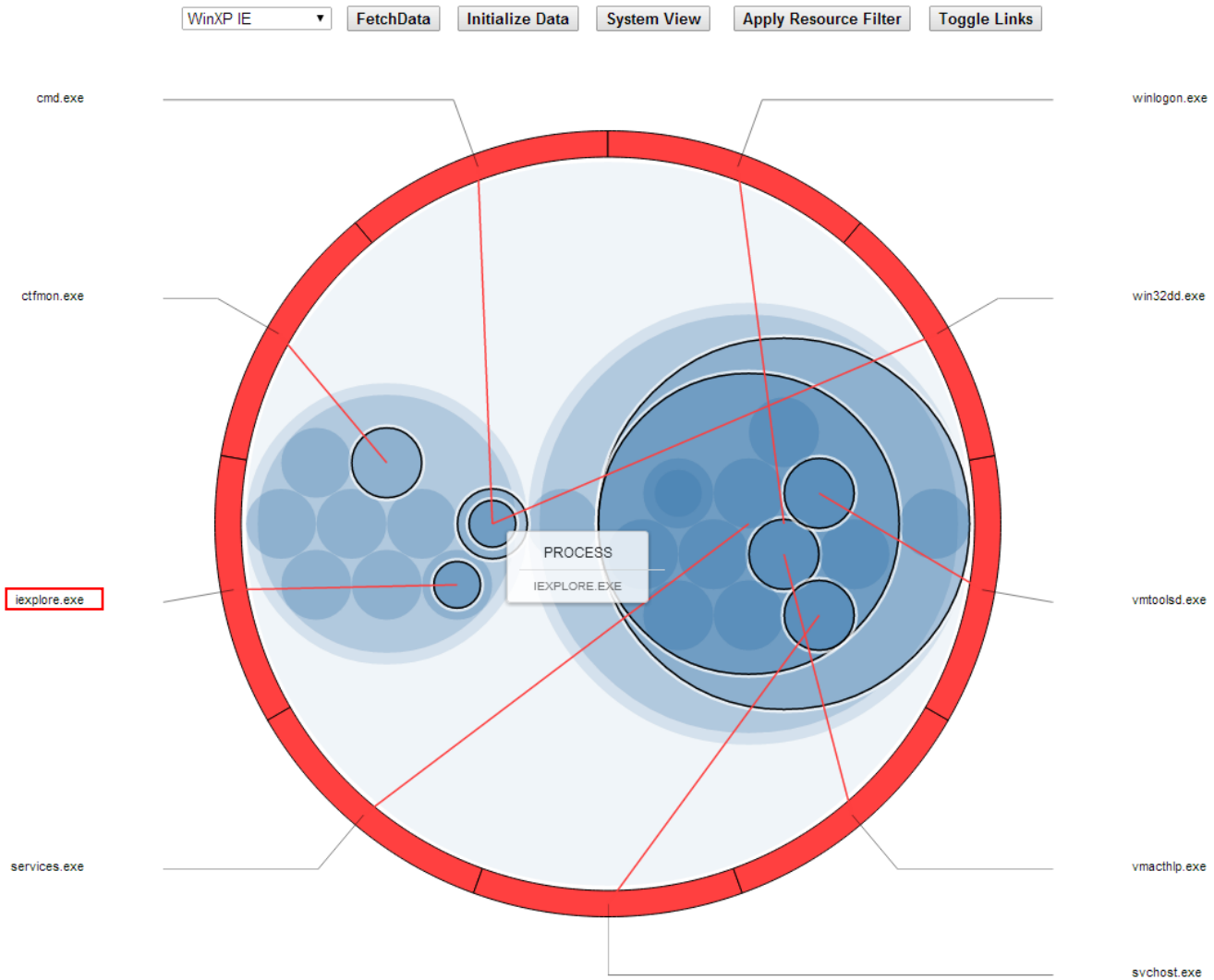

Figure 26. IE – Loaded Modules (.exe) Links.

4.3.2 Windows XP SP3 (Google Chrome) – Trusted Image

The second memory image is from a Windows XP machine executing a single instance of Google Chrome. Applying the unique resource filter provides a list of connections (ports and sockets), user files, and loaded modules. Clicking on the Socket and Port Resource arcs populates the resource ring with the respective resource lists. Using method 1 and clicking the *Toggle Links* button when viewing each list provides the user with unique visualizations (see Figures 27, 28). The cluster of `CHROME.EXE` nodes can be identified by three socket connections, as well as the only three active open ports on the system.

Filtering the global resource list by clicking the *Apply Resource Filter* button and selecting the red loaded modules arc slice populates the resource ring with the list of executable loaded modules. Toggling the links for this list provides another view (see Figure 29). By browsing this short list the user can identify the `chrome.exe` module and verify that it is loaded by the appropriate `CHROME.EXE` process node as indicated by the highlighted node cluster.

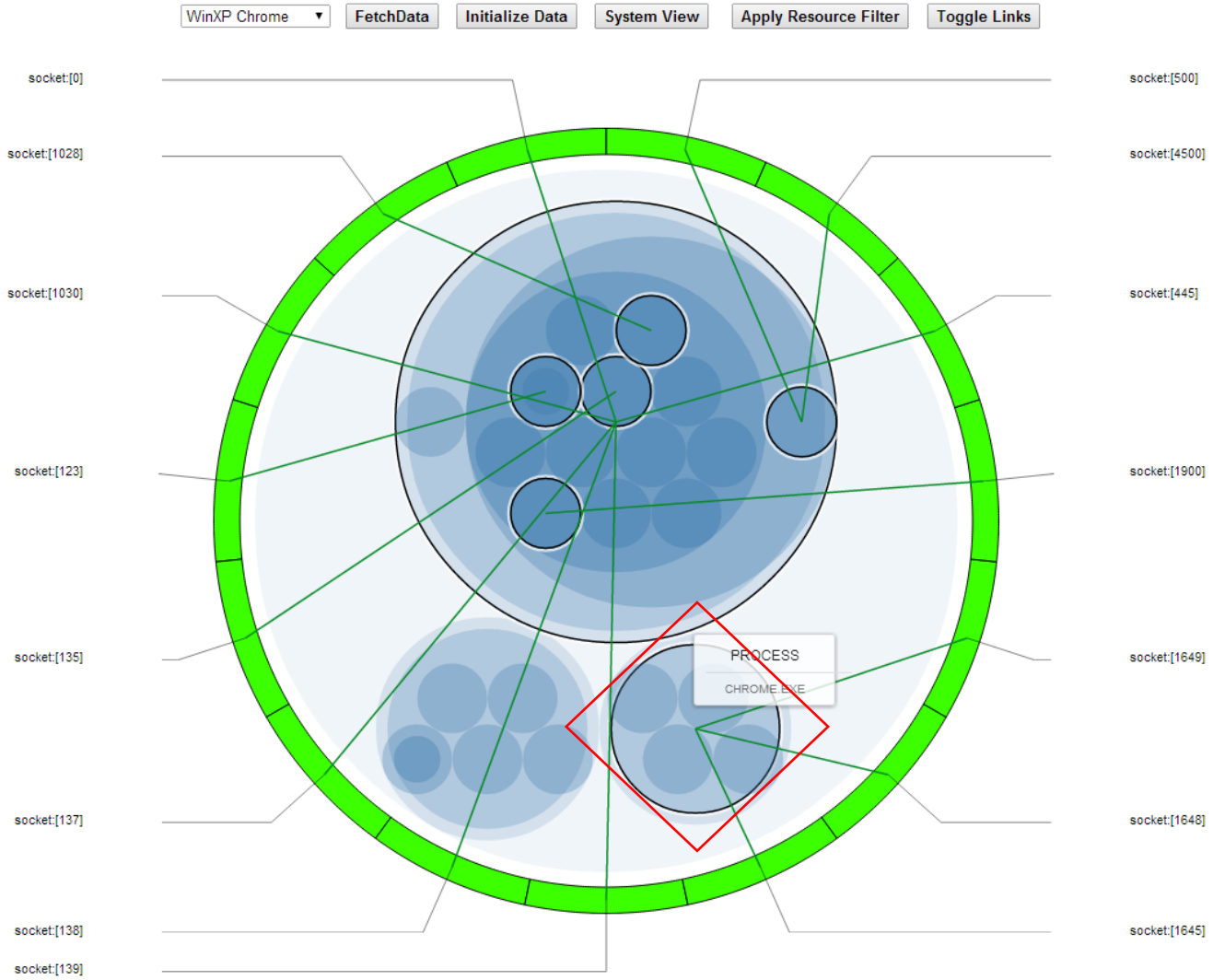

Figure 27. Chrome – Sockets List Links.

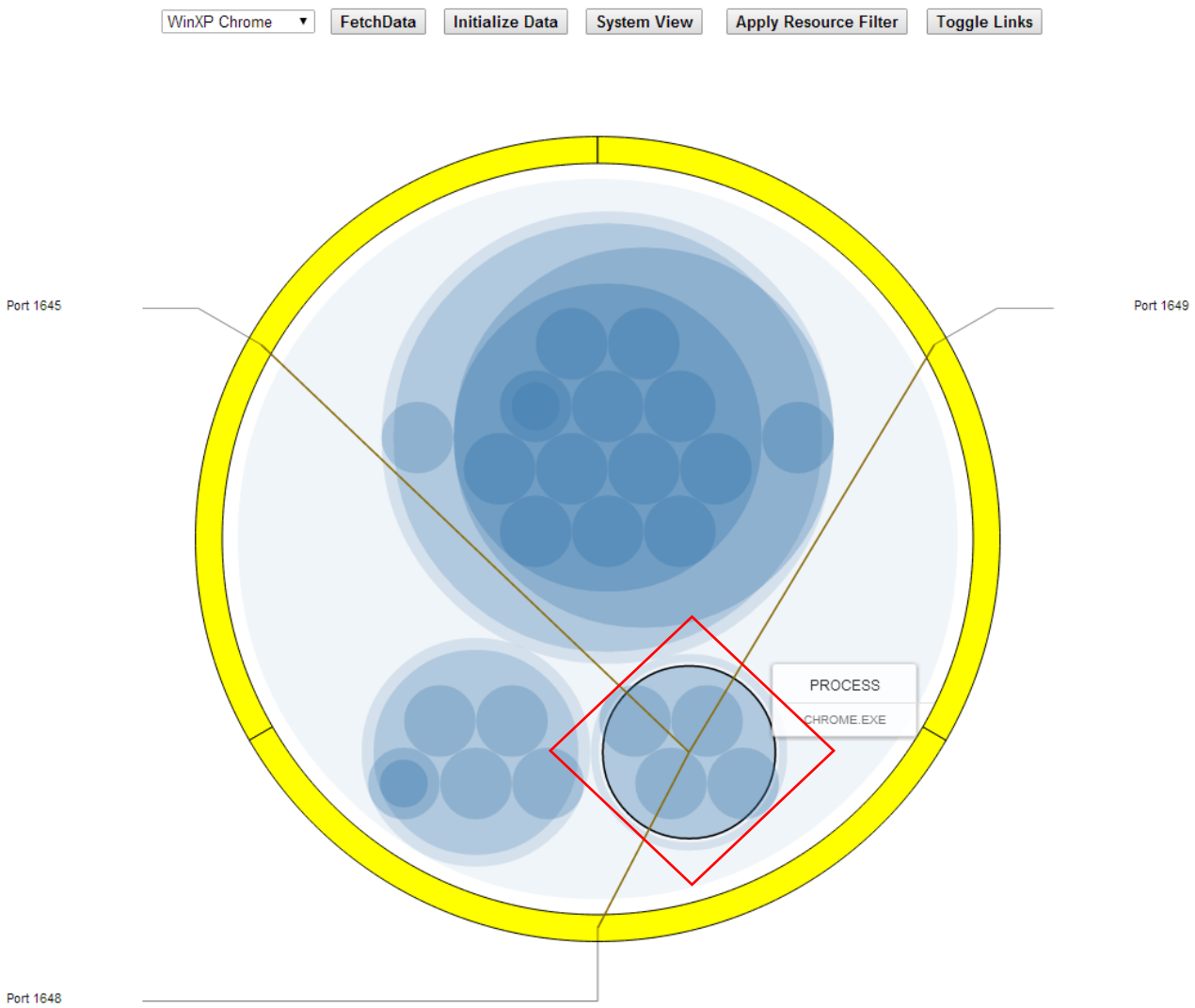

Figure 28. Chrome – Ports List Links.

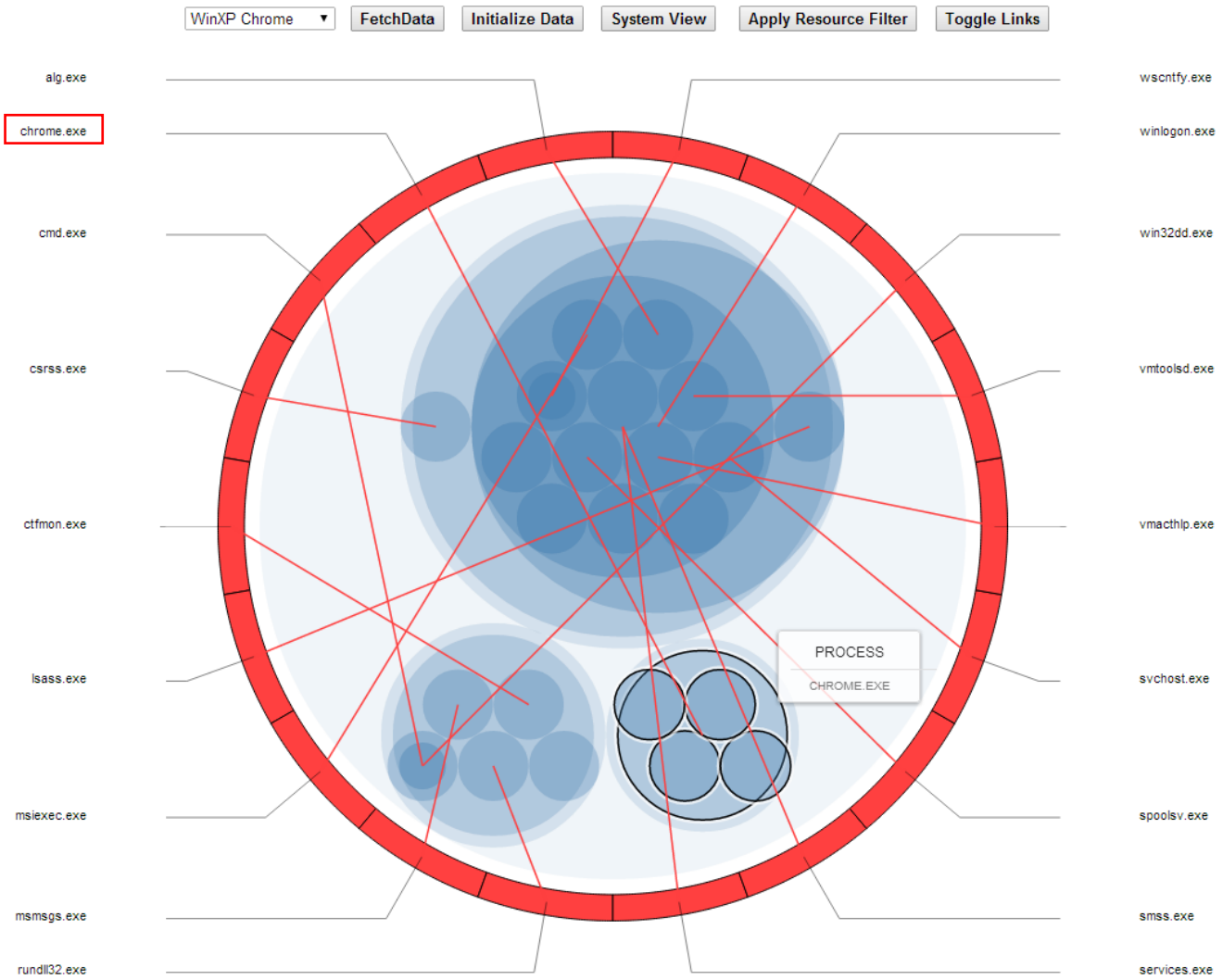

Figure 29. Chrome – Loaded Modules (.exe) Links.

4.3.3 Windows XP SP3 (Mozilla Firefox) – Trusted Source

The third trusted memory dump is from a Windows XP machine executing a single instance of Mozilla Firefox. Applying the unique resource filter provides a list of connections (ports and sockets), user files, and loaded modules. Clicking on the Socket and Port Resource arcs populates the resource ring with the respective resource lists. Applying method 1 and clicking the *Toggle Links* button when viewing each list provides the user with unique visualizations (see Figures 30, 31). An instance of `PLUGIN-CONTAINER.EXE` is running inside a `FIREFOX.EXE` process node and can be identified by eleven socket connections, as well as the only eleven active open ports on the system.

Filtering the global resource list by clicking the *Apply Resource Filter* button and selecting the red loaded modules arc slice populates the resource ring with the list of executable loaded modules. Toggling the links for this list provides another view (see Figure 32). By browsing this short list the user can identify the `firefox.exe` and `plugin-container.exe` modules and verify that they are loaded by the appropriate `FIREFOX.EXE` and `PLUGIN_CONTAINER.EXE` process nodes as indicated by the highlighted nodes.

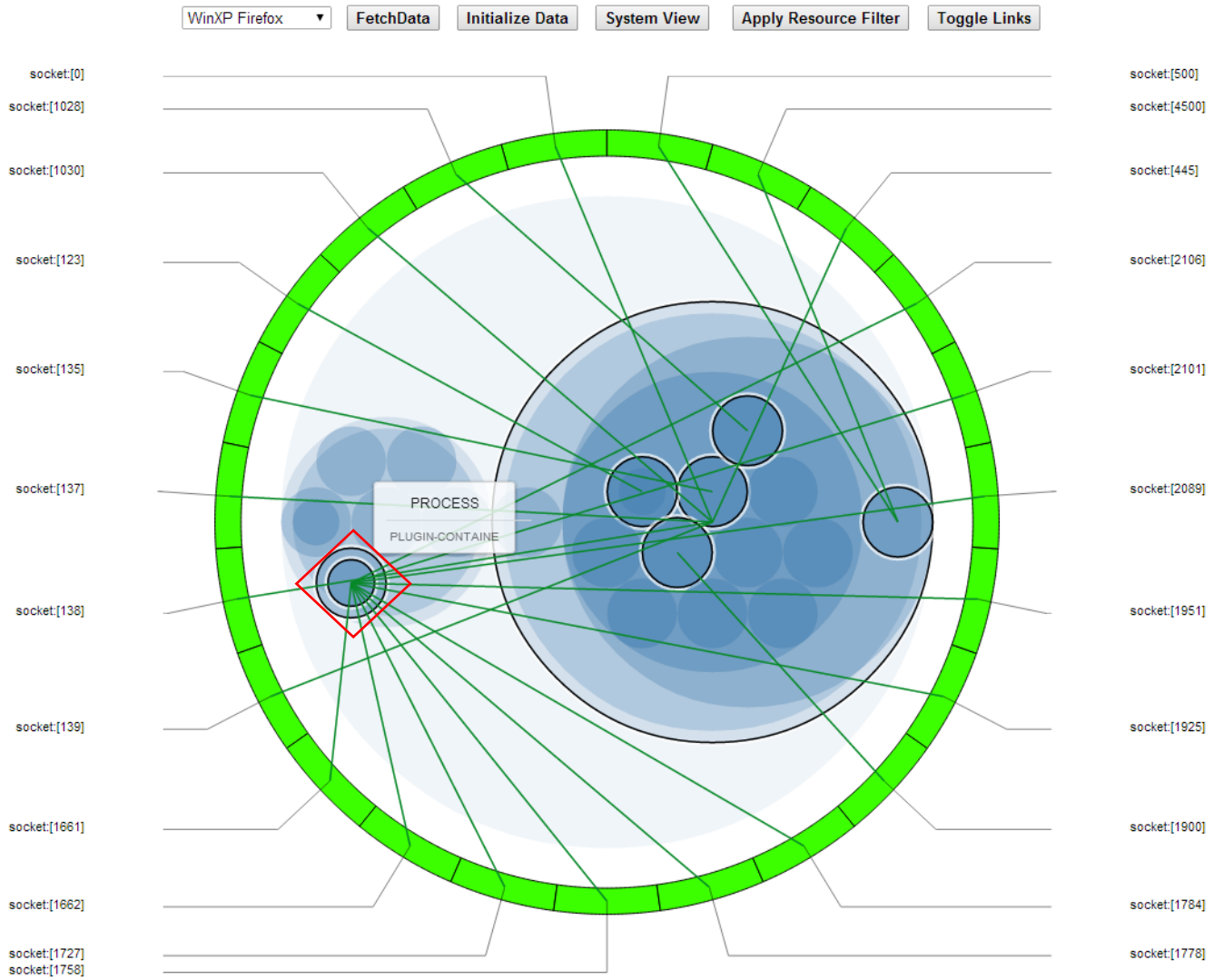

Figure 30. Firefox – Socket List Links.

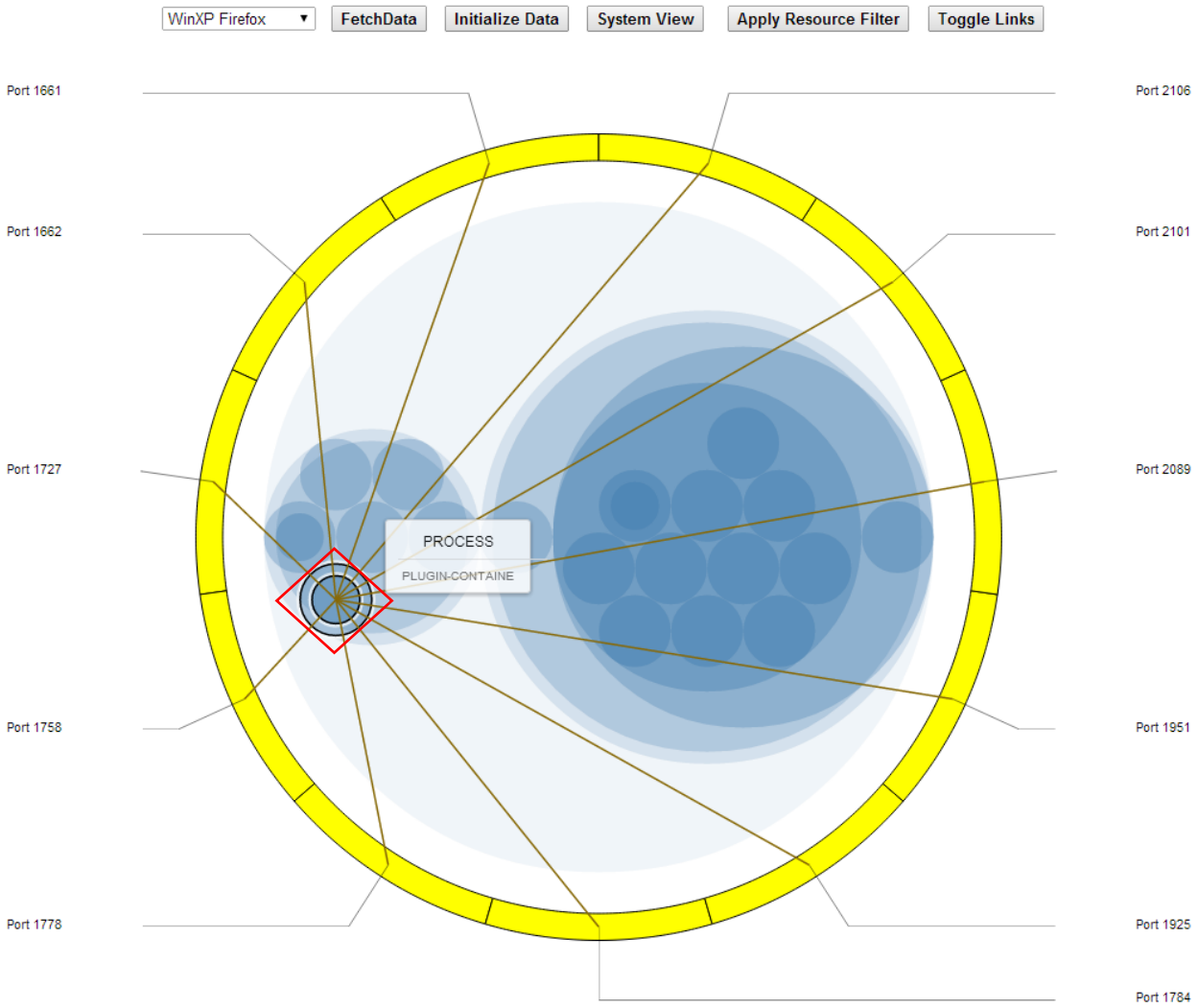

Figure 31. Firefox – Port List Links.

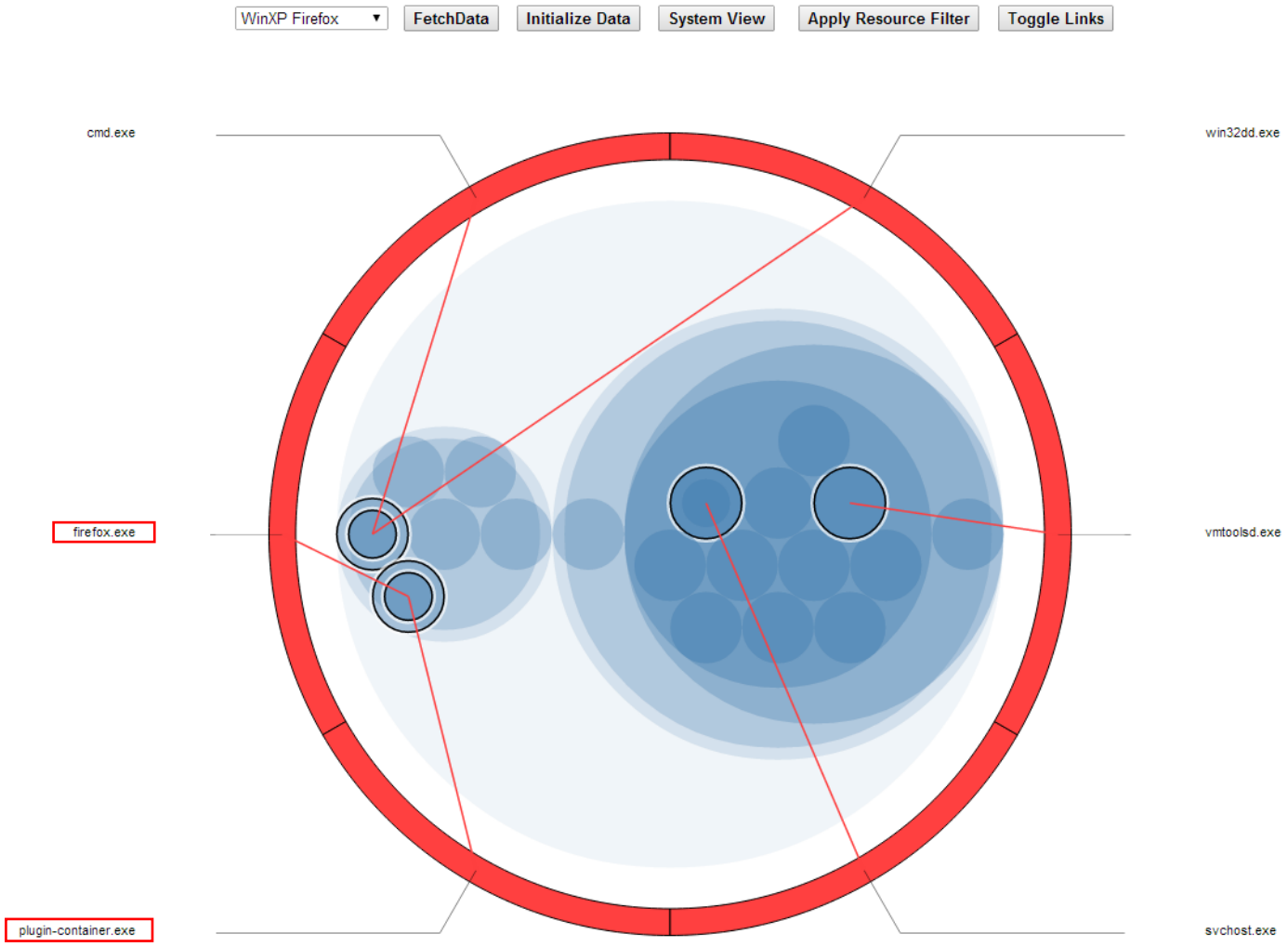

Figure 32. Firefox – Loaded Modules (.exe) Links.

4.3.4 Windows XP SP3 (Solitaire) – Trusted Source

The fourth trusted memory image is from a Windows XP machine executing a single instance of Solitaire. Again the unique resource filter provides a list of connections (ports and sockets), user files, and loaded modules. Clicking on the Socket Resource arc populates the resource ring with the respective resource list. Applying method 1 and clicking the *Toggle Links* button when viewing the socket list provides the user with the visualization in Figure 33, however, none of the sockets are connected to the `SOL.EXE` process. There are no open ports and therefore no associated unique visualization.

Filtering the global resource list by clicking the *Apply Resource Filter* button and selecting the red loaded modules arc slice populates the resource ring with the list of executable loaded modules. Toggling the links for this list provides another view (see Figure 34). By browsing this short list the user can identify the `sol.exe` module and verify that they it is loaded by the appropriate `SOL.EXE` process as indicated by the highlighted node.

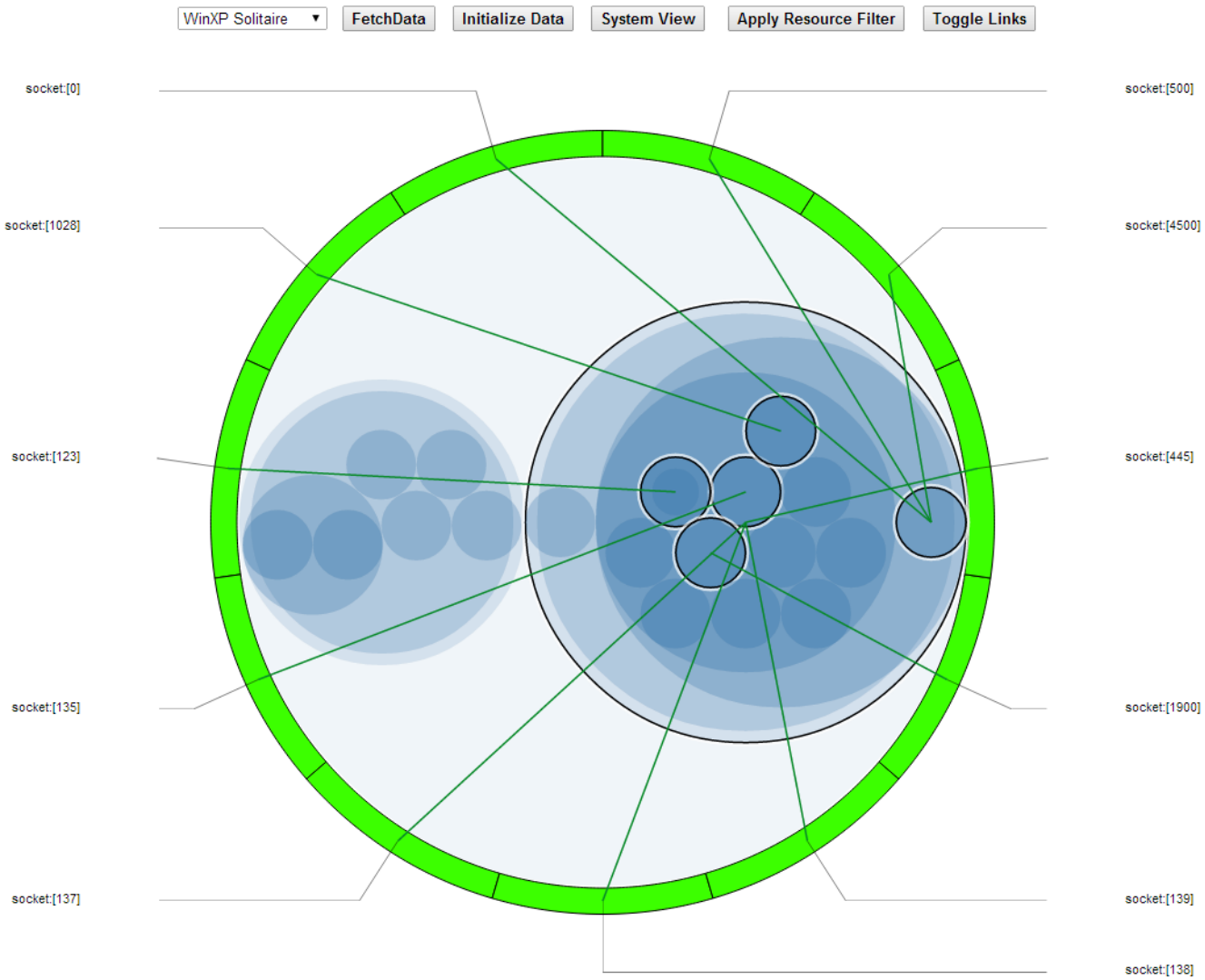

Figure 33. Solitaire – Socket List Links.

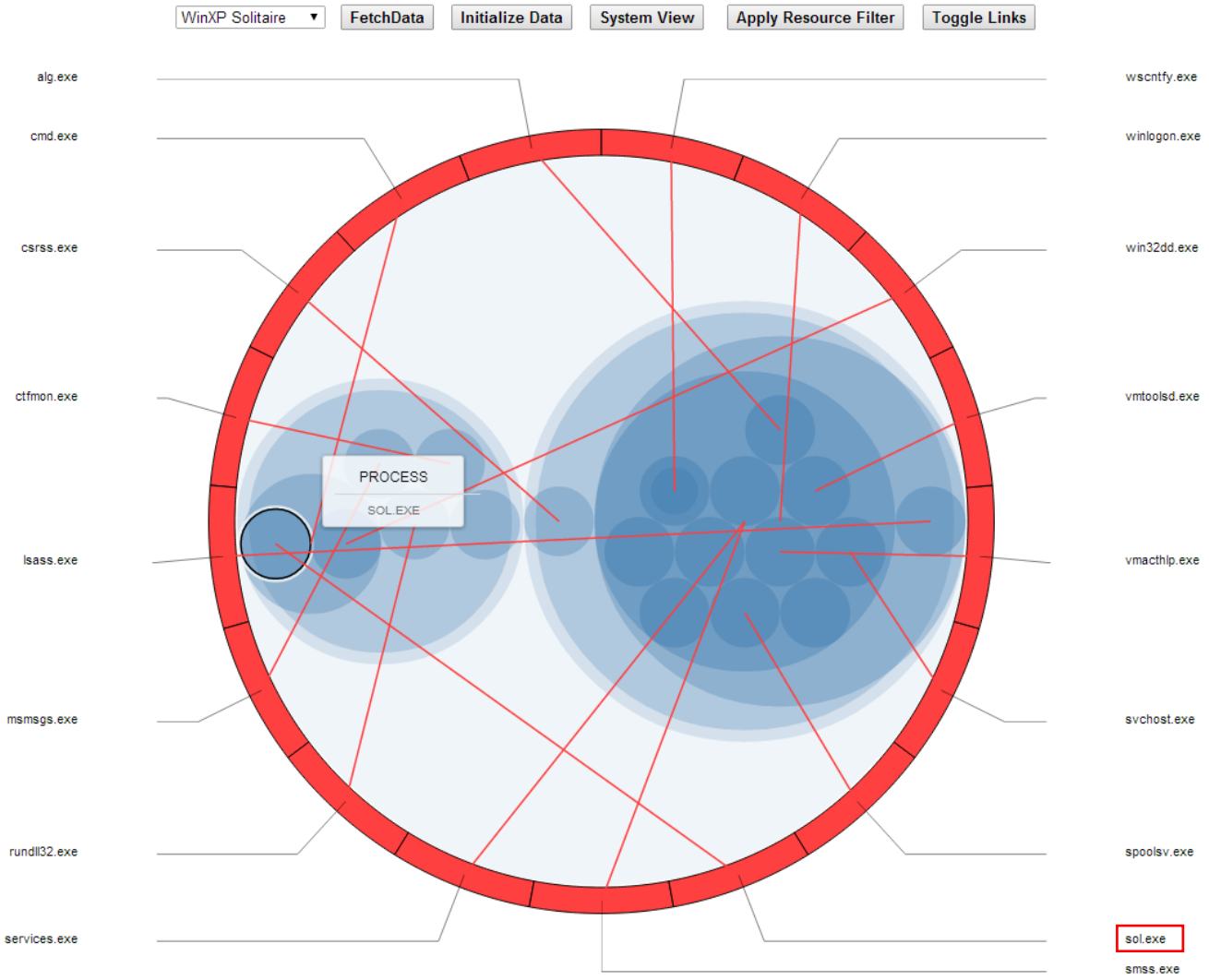

Figure 34. Solitaire – Loaded Modules (.exe) Links.

4.3.5 Windows XP SP3 (Word) – Trusted Source

The fifth trusted memory image is from a Windows XP machine executing a single instance of Microsoft Word. The unique resource filter provides a list of sockets, user files, and loaded modules. There are no open ports in this dataset. The Word process is discovered by exploring the node layout as seen in Figure 35. Applying method 1 and clicking the *Toggle Links* button when viewing the files list provides the user with the visualization in Figure 36. This visualization lists many items including the open document (.docx), a Word template (Normal.dotm), a picture file (.jpeg), and various temp files. The combination of the .docx file residing in the users directory and the temp files reflects the fact that the document was in the process of being edited at the moment of memory capture.

One of the objectives for digital forensic practitioners is to gain an insight into the specific user activity on a target device through analysis of the collection data. This particular test dataset provides an example of how the interactive visualization could prove useful to an analyst in identifying specific document editing activity. The same technique could be used to locate other instances of user activity such as recent web browsing, email editing, social networking, peer-to-peer file exchanges, and multimedia viewing. This information is valuable to digital forensic investigators looking to develop an activity profile of the system user.

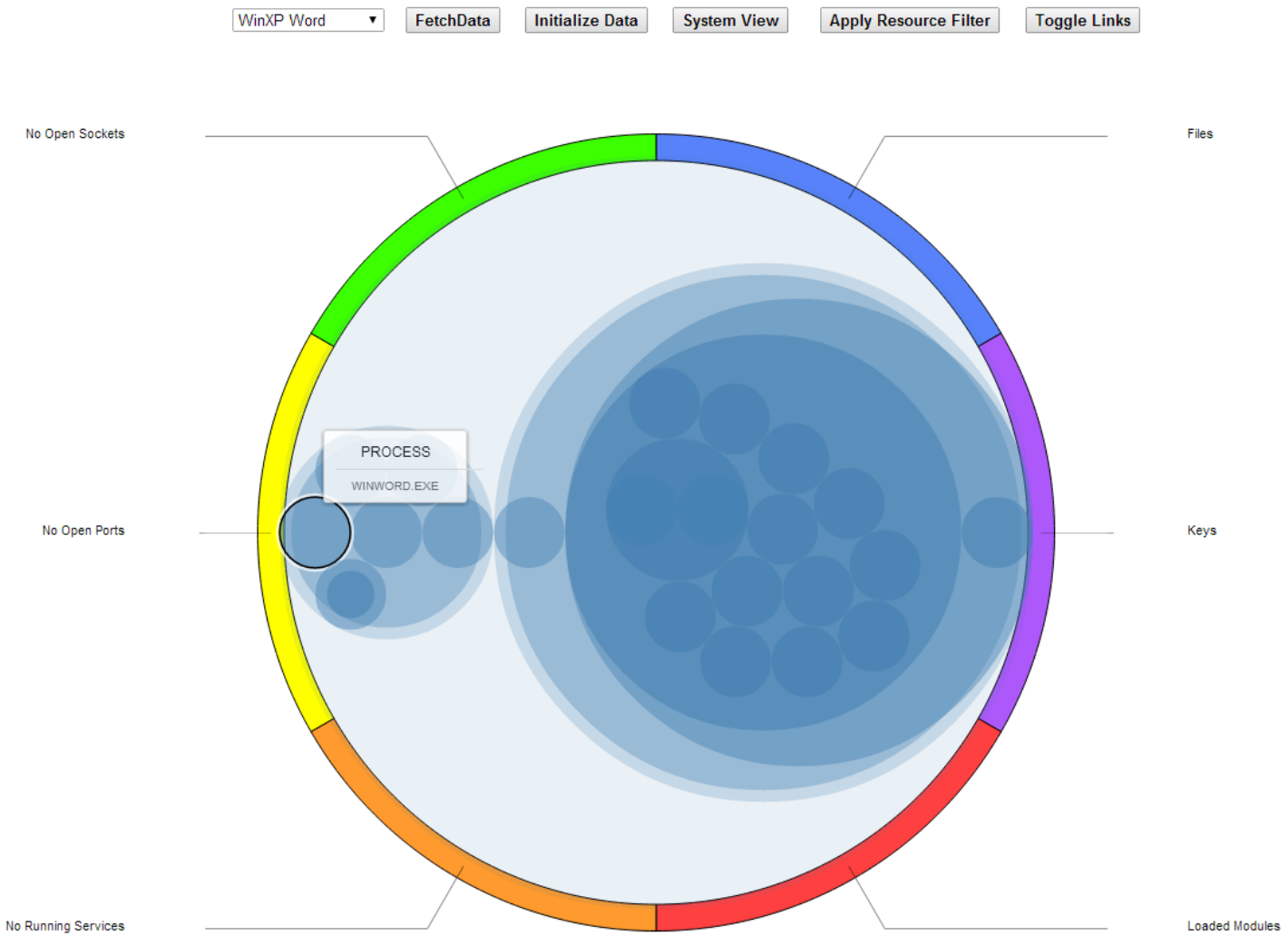

Figure 35. Word - Process Highlighted.

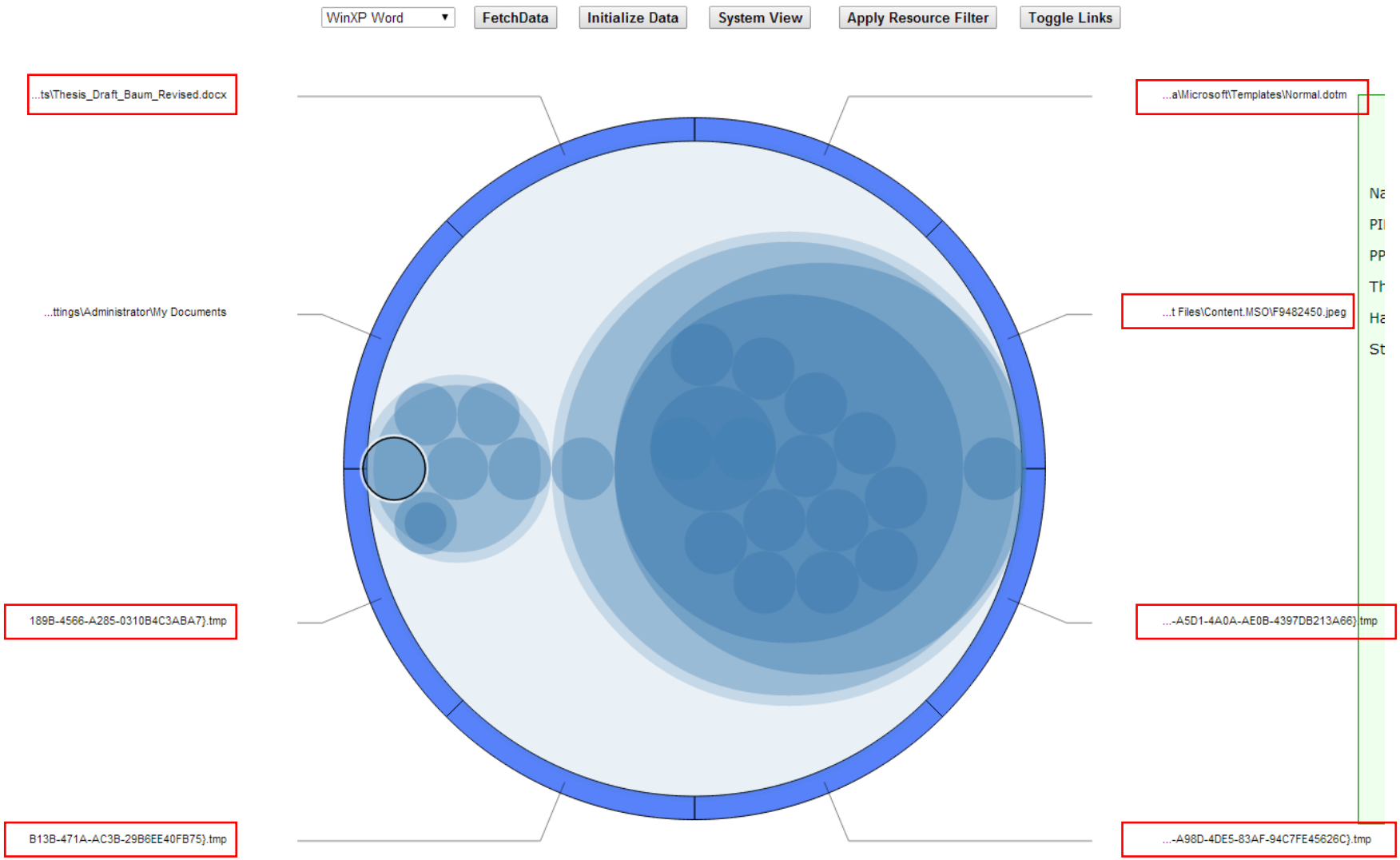

Figure 36. Word – Files List.

4.3.6 Windows XP SP3 (Malware 1)

The sixth memory dump tested is from a Windows XP machine executing an instance of the FUTo malware. The unique resource filter provides a list of connections (ports and sockets), user files, and loaded modules. Clicking on the Socket Resource arc populates the resource ring with the respective resource list. Applying method 1 and clicking the *Toggle Links* button when viewing the socket list provides the user with the visualization in Figure 37. Although examination does not provide any immediate observations, comparison with previous trusted examples reveals a similarity with the Solitaire image in the socket list link layout pattern (see Figures 33, 37). Closer inspection uncovers the only difference, namely, the absence of a `SOL.EXE` process node in the malware sample. There are no open ports and therefore no associated unique visualization.

The FUTo malware (discussed in Section 3.3.2) hides an active process by removing references to it in various kernel data structures. Since the process list extracted from memory is generated by walking these data structures, the hidden process will not appear as a node and instead exist as a loaded module attached to a system process. A review of the loaded modules for the `System Idle Process` reveals a sol.exe module connected to it (see Figure 38). Under normal circumstances, the `System Idle Process` would not load the Microsoft Solitaire program as a module. From this verification, there is sufficient evidence to categorize this as a potential implementation of a rootkit.

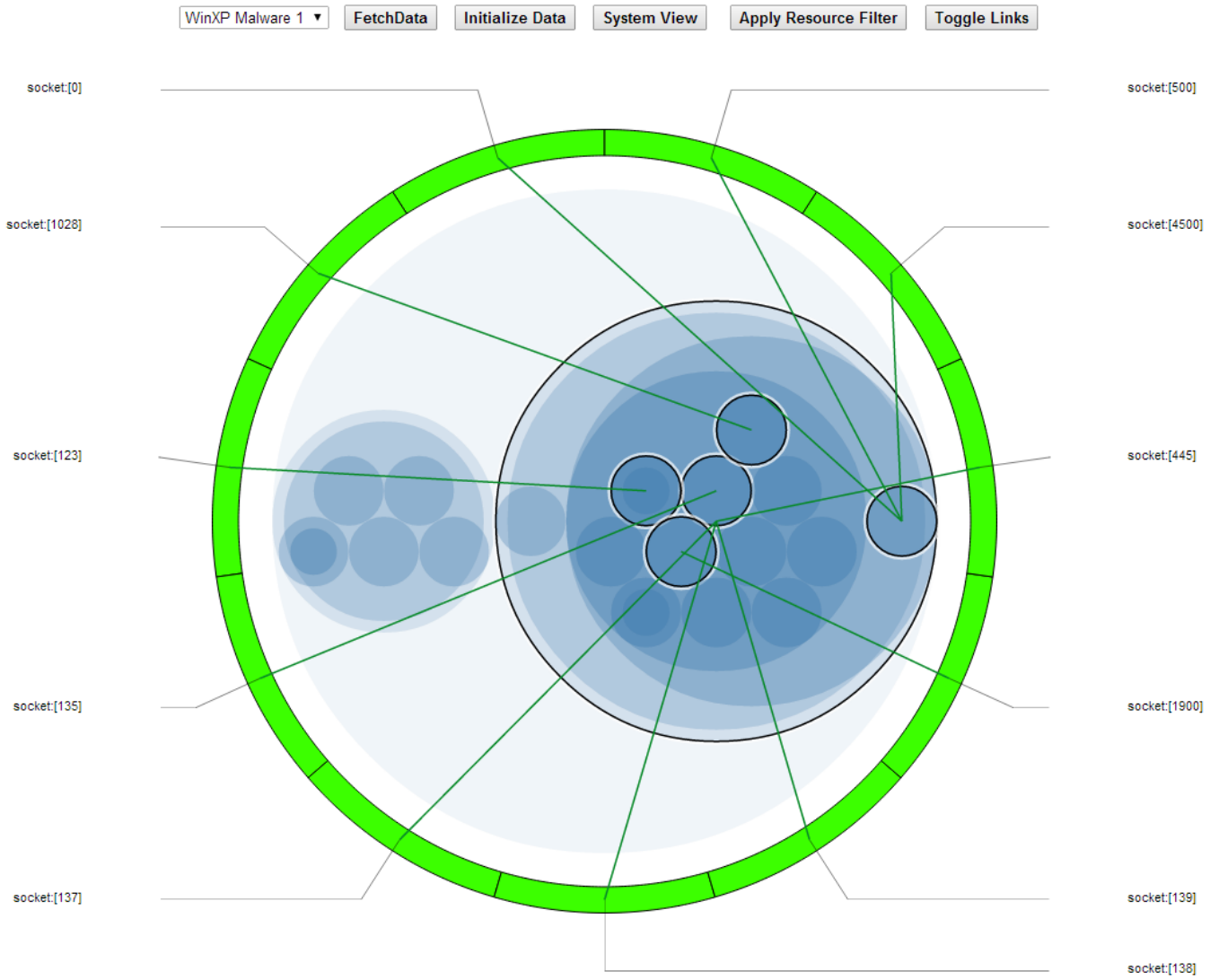

Figure 37. Malware 1 – Socket List Links.

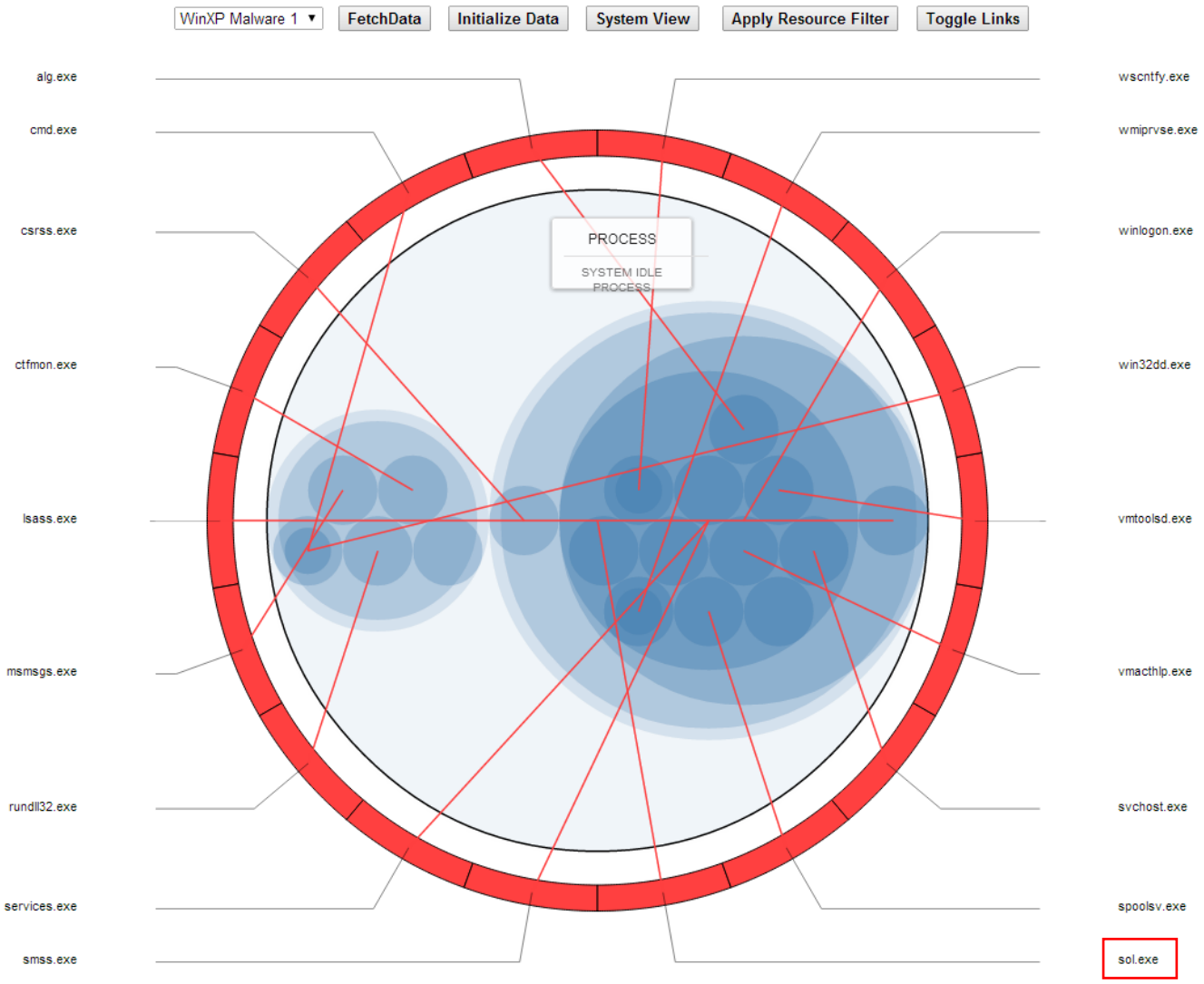

Figure 38. Malware 1 – Loaded Modules (.exe) Links.

4.3.7 Windows XP SP3 (Malware 2)

This seventh dump image also has some form of malware operating on the system at the time of memory acquisition. Applying the unique resources filter and toggling the links for each resource list provides the views shown in Figures 39, 40, and 41 respectively.

Examination of the socket list with links does not seem to display any unique patterns or trends visually. The port list with links shows one active connection to a process named `IEXPLORE.EXE`.

If method 2 is applied, this dataset should be compared to a similar trusted image. Since a process named `IEXPLORE.EXE` is listed it is compared to the trusted IE dataset from the previous section.

A few observations can be made:

- The single connection to `IEXPLORE.EXE` in the malware dataset differs from the numerous connections seen in the trusted dataset (see Figures 24, 25).
- The iexplore.exe loaded module that appears in the trusted dataset appears differently in the Malware 2 set as `IEXPLORE.EXE` (see Figure 26).

Using these observations it can be deduced that the malware in this dataset is probably masquerading as an instance of IE. In Section 3.3.2, the Poison Ivy Remote Administration Tool (RAT) is discussed as a form of malware. A key behavior of Poison Ivy is its ability to inject itself into a browser process. Considering the single connection to this process we can classify this as a potential RAT signature.

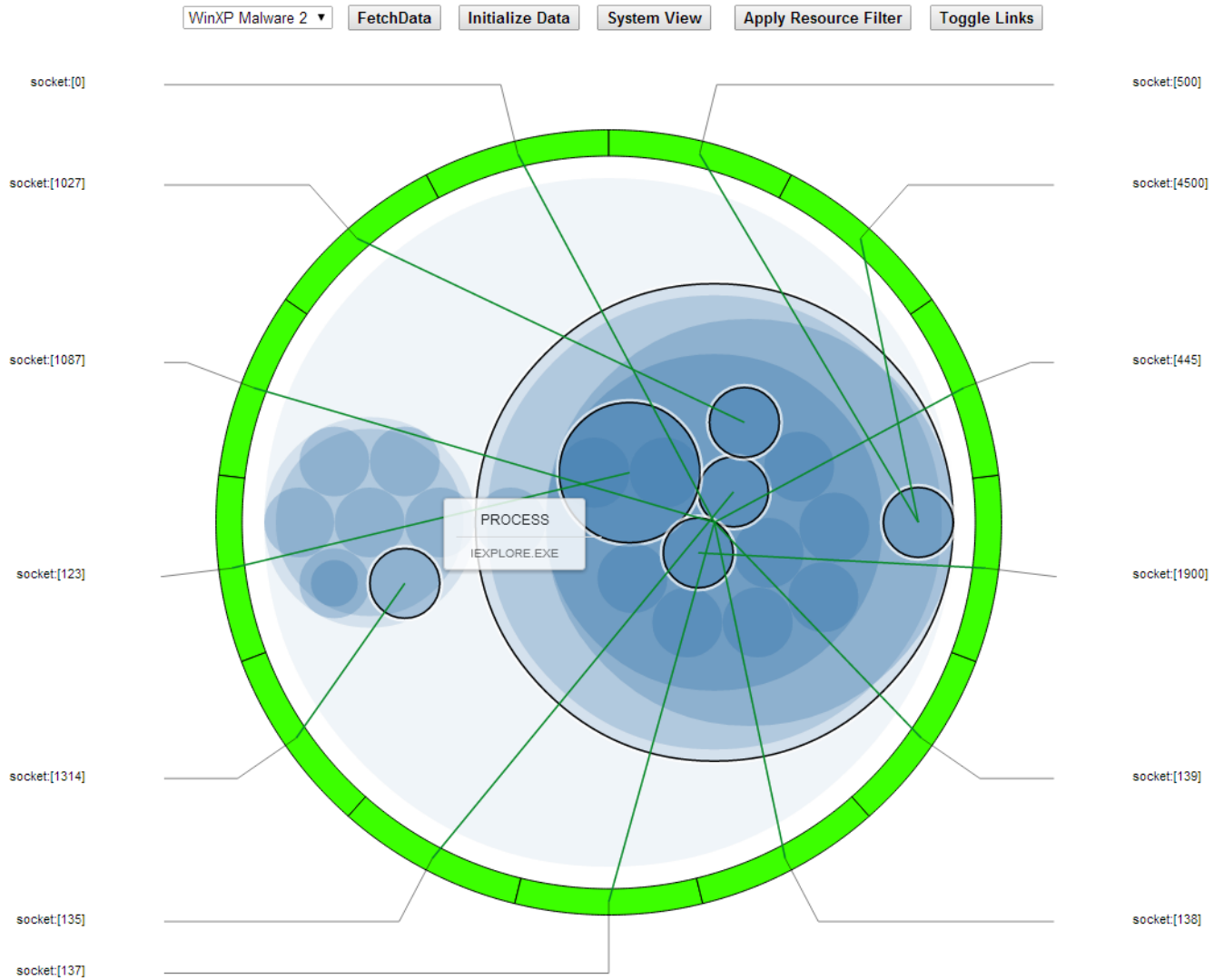

Figure 39. Malware 2 – Socket List Links.

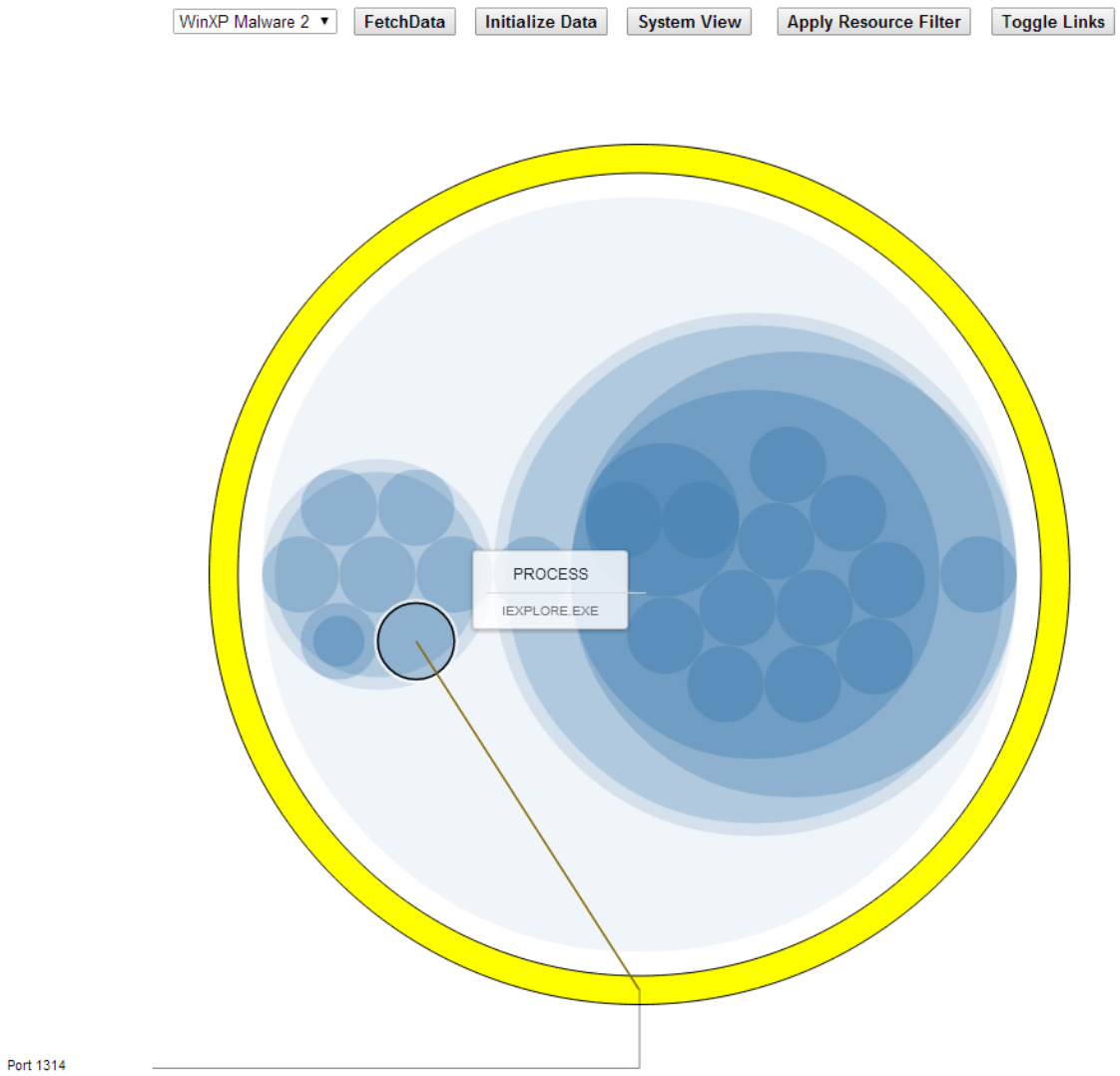

Figure 40. Malware 2 – Port List Links.

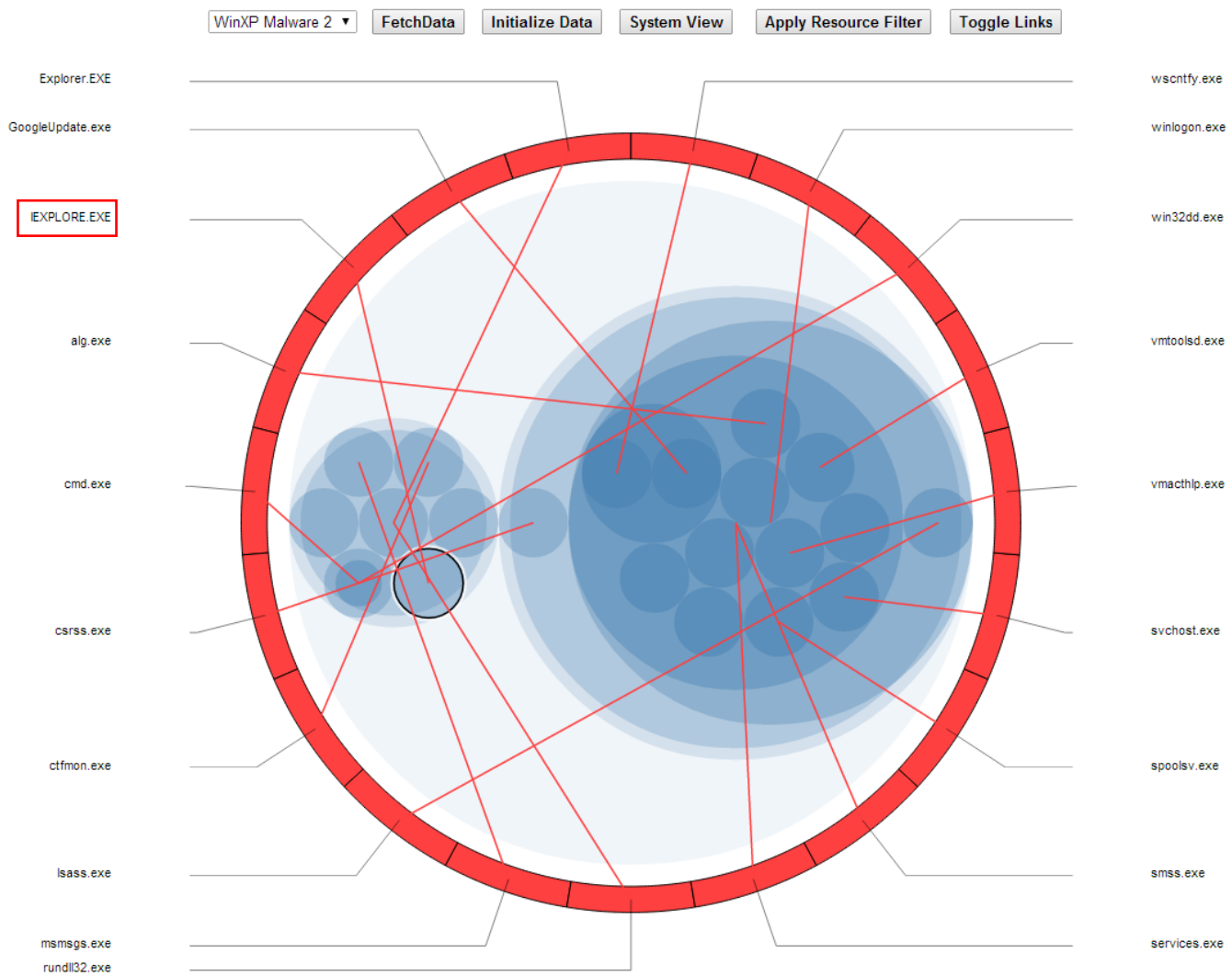

Figure 41. Malware 2 – Loaded Modules (.exe) Links.

4.3.8 Windows XP SP3 (Malware 3)

The final forensic memory image also contains malware operating on the system at the time of memory acquisition. Filtering for unique resources and toggling the links for each resource list provides the views shown in Figures 42, 43, and 44.

Examination of the socket list visualization with links does not seem to display any unique patterns or trends. It does bear a close resemblance to the socket layout from the Malware 2 dataset (see Figure 39). The ports list, however, shows only one connection which is attached to an msmsgs.exe process node. The `MSMSGS.EXE` process appears in multiple datasets and appears to be a legitimate process but does not have any connections in any of the trusted samples (see Figure 45 from trusted Solitaire dataset). This observation coupled with the single connection reveals a pattern that can be treated as a possible instance of RAT malware similar to the Malware 2 dataset

Figure 42. Malware 3 – Socket List Links.

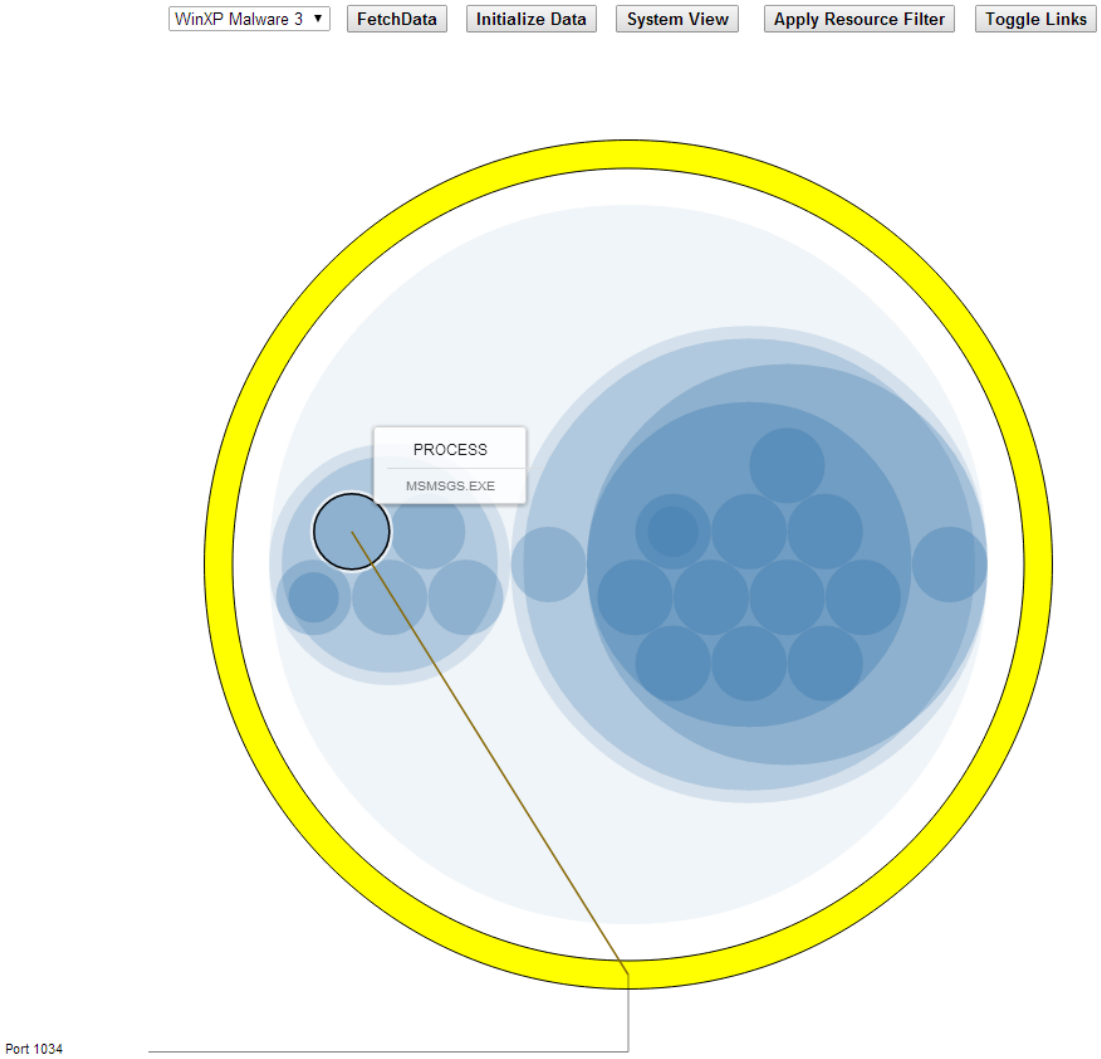

Figure 43. Malware 3 – Port List Links.

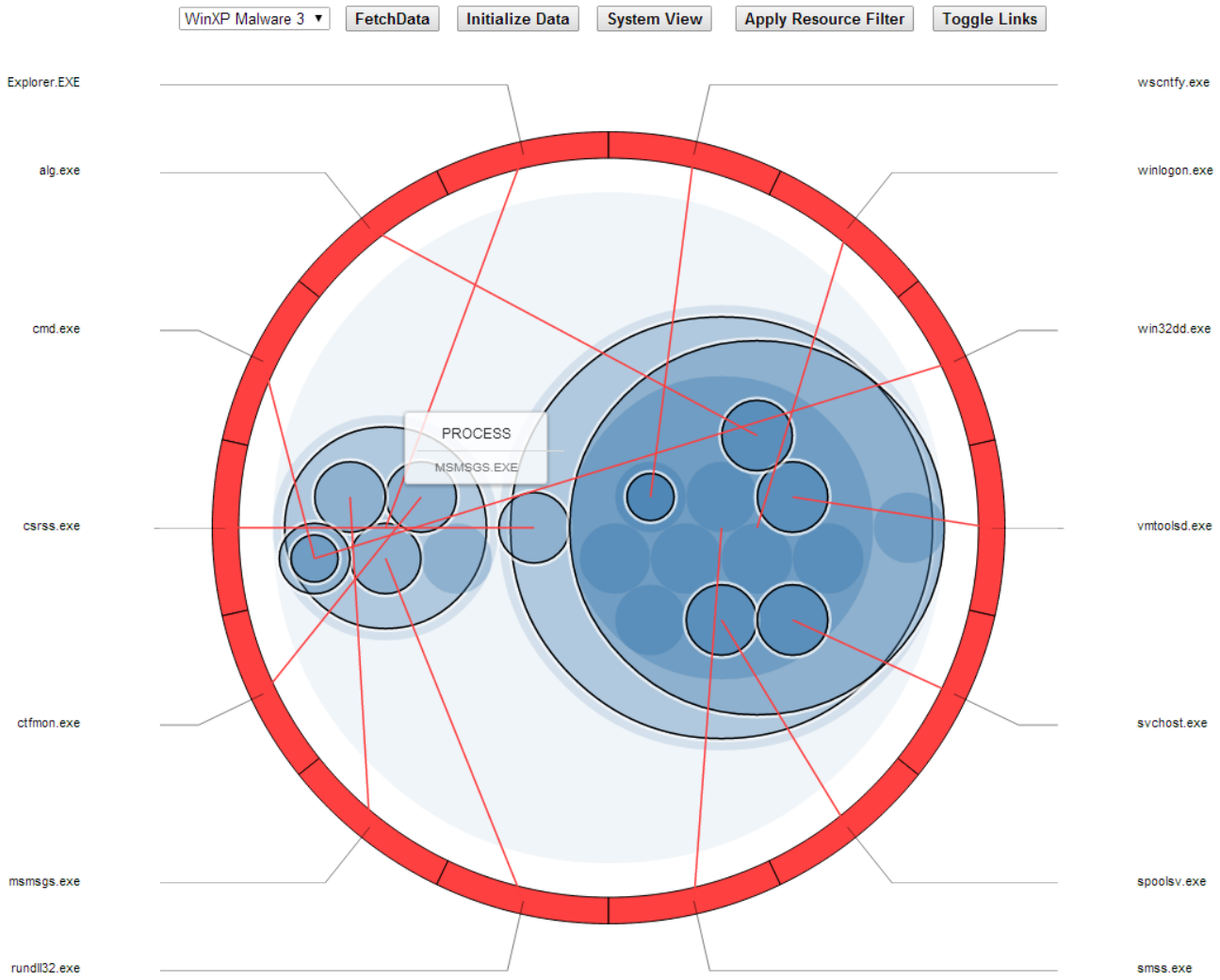

Figure 44. Malware 3 – Loaded Modules (.exe) Links.

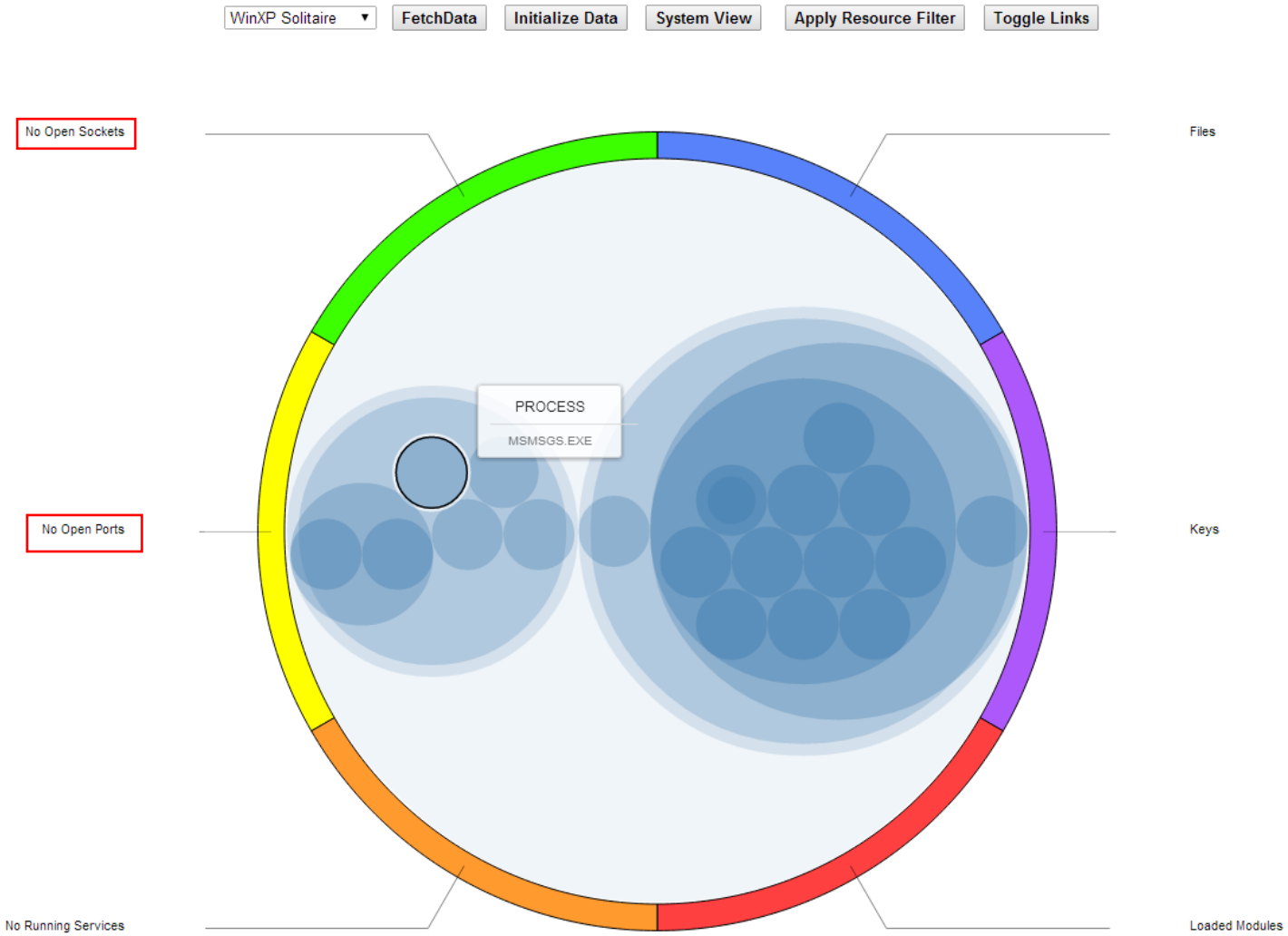

Figure 45. Solitaire – `MSMSGS.EXE` process highlighted.

4.4 Summary

This chapter applies the D3 JavaScript Visualization tool to various test datasets extracted from a number of Windows XP memory dumps. The results from the eight sample datasets are evaluated against the three research goals outlined in Chapter 1. The first objective requires the developed visualization tool provide a global view of the data that is extracted from forensic memory dumps using memory analysis tools. The second goal expects the tool to possess the ability to filter datasets and highlight two key relationships within the data. The first is the relationship between a process and its associated resources, while the second is the relationship between a particular resource and its corresponding process nodes. The final objective of the research requires the visualization tool contain features that assist the user in locating behavior patterns and unique items of relevance within the datasets.

Five of the Windows memory captures contain legitimate (trusted) process activity while three datasets contain malware. Evaluation of the trusted datasets provides profiles of legitimate process activity that can be compared against malware datasets. By applying the features of the visualization tool, FUTo malware activity is located in the Malware 1 dataset while instances of the Poison Ivy Remote Administration Tool are detected in both the Malware 2 and Malware 3 datasets.

V. Conclusions and Recommendations

The proof-of-concept visualization tool developed in this research is shown to provide a level of assistance to forensic investigators in the discovery of new data and existing patterns. The specific objectives for this research outlined in Chapter 1 are as follows:

1. The visualization tool should provide an interactive global view of the data output from forensic memory image analysis tools.
2. The visualization tool should have the ability to filter the datasets for a particular system and visually represent the following associations:
    - The relationship between each process and its associated resources.
    - The relationship between a system resource and its associated processes.
3. The visualization tool should assist the user in locating unique patterns and information of relevance within the datasets.

By addressing the overall problem statement and meeting the research objectives listed above, it can be implied that the thesis research is successful.

5.1 Research Accomplishments

This research obtained forensic memory images from Windows XP machines and developed an interactive software tool that graphically represents the extracted datasets using Information Visualization techniques. Multiple systems containing forms of malware were analyzed using the developed visualization tool to discover specific behavior patterns and attempt to identify the type of malware. The visualization tool

created using D3 JavaScript is capable of running on virtually any platform due to its web-based design making it completely portable for use in the field.

5.2 Future Work

One shortcoming of the visualization tool developed in this research is its limitation to a single data input format. The JavaScript source code for this project is specifically tailored to import data files that are in a particular format (CSV). One web blog describes a method of using an SQL database to generate datasets based on queries (D3noob, 2013). Streamlining both the collection and digital artifact extraction processes into one automated routine would improve the overall usability and increase the speed of analysis.

One potential method of improving the visualization and location of items of interest in large datasets could involve the whitelisting of trusted processes and resources. Placing less emphasis on trusted objects may help digital forensic analysts focus on the important data contained in a particular collection.

A useful addition to the visualization tool would involve the automatic detection of hidden processes similar to the FUTo and RAT malware discovered in Chapter 4. By understanding the behavior of certain process hiding mechanisms, the tool could filter and search the data to identify such cases and highlight the responsible process nodes and resources respectively.

While the eight datasets tested in this research utilize the Window XP operating system, the visualization tool can also be applied to datasets from Windows 7 forensic memory images as shown in Figure 46. CMAT is used to perform the extraction of digital

artifacts from the dump due to the limited number of Volatility plugin modules currently compatible with Windows 7 architecture.

The digital forensic visualization methods used in this research can also be ported to various operating systems including Mac OS X, Linux, and Android. A prototype model for an Android 4.3 memory dump was developed during the course of this research but includes a limited feature set. Figure 47 contains a screen capture of the Android process node layout. Since both Android and Linux are fairly similar in architecture, it would appear fairly simple to modify the system to accept datasets from both operating systems.

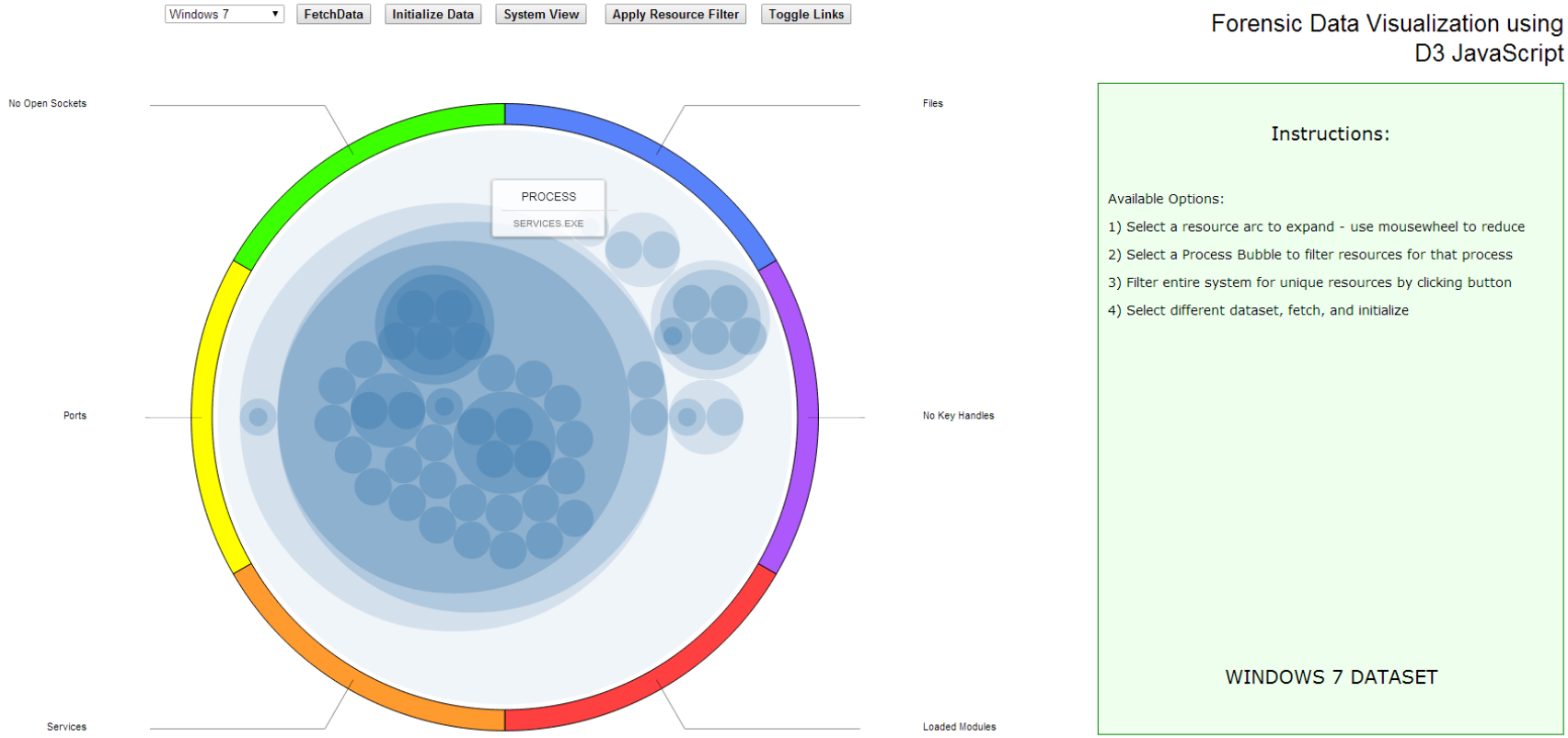

Figure 46. Windows 7 Dataset Visualization.

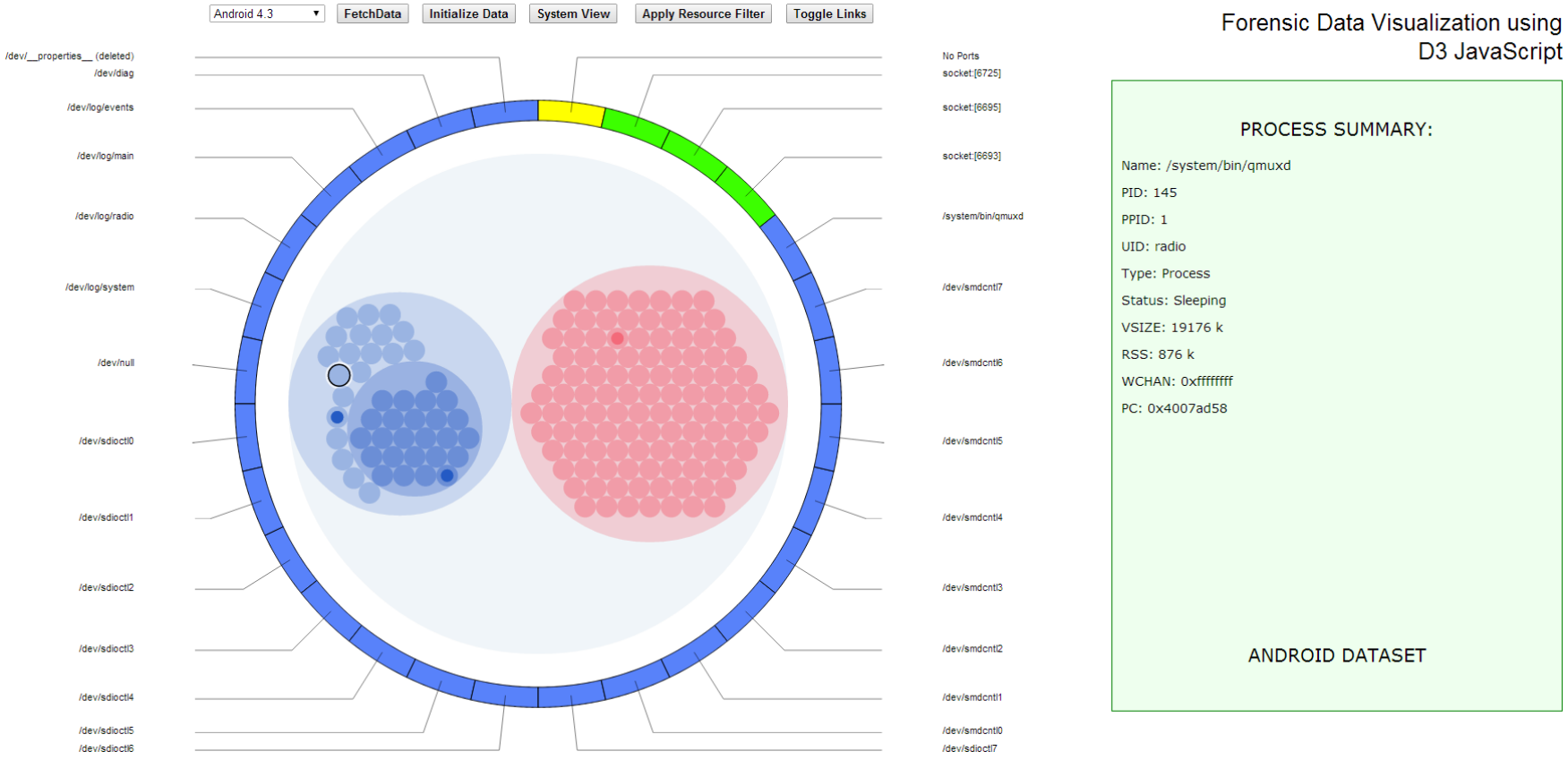

Figure 47. Android 4.3 Dataset Visualization (Prototype).

Bibliography

Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An Open Source Software for Exploring and Manipulating Networks. *Proceedings of the Third International ICWSM Conference,* 361-362.

Beebe, N., & Clark, J. (2005). Dealing With Terabyte Data Sets In Digital Investigations. *Advances in Digital Forensics: IFIP International Conference on Digital Forensics*, vol. 194, 3-16.

Bennett, J., Moran, N., & Villeneuve, N. (2013). FireEye Poison Ivy Report. http://www.fireeye.com/resources/pdfs/fireeye-poison-ivy-report.pdf

Bostock, M. (2014). D3 Documentation Wiki. *Internet.* https://github.com/mbostock/d3/wiki

Brightpoint Inc. (2014). Political Influence D3 Visualization. *Internet.* http://www.brightpointinc.com/interactive/politicalinfluence/index.html?source=d3js

Burdach, M. (2006). Physical Memory Forensics [slides]. *Black Hat Federal*. https://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Burdach.pdf

Cai, L., Sha, J., & Qian, W. (2013). Study on Forensic Analysis of Physical Memory. *2nd International Symposium on Computer, Communications, Control, and Automation*, 221-224.

Carrier, B., & Grand, J. (2004). A Hardware-Based Memory Acquisition Procedure for Digital Investigations. *Journal of Digital Investigations,* 1(1), 50-60.

Carroll, O., Brannon, S., & Song, T. (2008). Computer Forensics: Digital Forensic Analysis Methodology. *Computer Forensics,* 56(1), United States Department of Justice. http://www.justice.gov/usao/eousa/foia_reading_room /usab5601.pdf

Carvey, H. (2009). *Windows Forensic Analysis DVD Toolkit 2nd ed*. Burlington MA: Syngress.

D3noob. (2013, February 8). Using a MYSQL database as a source of data. Message posted to http://www.d3noob.org

Davis, N. (2008). *Live Memory Acquisition for Windows Operating Systems: Tools and Techniques for Analysis*. Thesis. Eastern Michigan University.

Faisal, S., Blandford, A., & Potts, H. W. (2013). Making sense of personal health information: Challenges for information visualization. *Health Informatics Journal*, 19(3), 198-217.

Federal Bureau of Investigation. (2014). Cyber Fact Sheet. *Cyber Crime*. http://www.fbi.gov/about-us/investigate/cyber/

Garfinkel, S. (2010). Digital forensics research: The next 10 years. *Digital Forensic Research Workshop (DFRW)*, 7(S), S64-S73.

Garner, G. (2013). Forensic Acquisition Tools. *Internet.* http://gmgsystemsinc.com/fau/

Gephi. (2014). The Open Graph Viz Platform. *Internet*. https://gephi.org

Henehan, B., Johas-Teener, M., Scholles, M., & Thompson, D. (2006). 1394 Standards and Specifications Summary [slides]. *1394 Trade Association*. http://www.1394ta.org/developers/specifications/StandardsOrientationV5.0.pdf

Keim, D. (2002). Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics.* 8(1), 1-8.

Kruse, W., & Heiser, J. (2002). *Computer Forensics: Incident Response Essentials*. Addison Wesley Publishing.

Liu, S., Cui, W., Wu, Y., & Liu, M. (2014). A Survey on Information Visualization: Recent Advances and Challenges. *The Visual Computer*, 30(1), 1-21.

Microsoft Corporation. (2014). Use the Microsoft Symbol Server to obtain debug symbol files. *Internet.* http://support.microsoft.com/kb/311503

NASDAQ Demo. (2014). Dimensional Charting JavaScript Library. *Internet.* http://nickqizhu.github.io/dc.js/

National Institute of Justice. (2008). *Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition*. Washington, D.C.

National Institute of Standards and Technology. (2007). *Guidelines on Cell Phone Forensics*. (NIST Special Publication 800-101). Gaithersburg, MD.

National Institute of Standard and Technology. (2006). *Guide to Integrating Forensic Techniques into Incident Response*. (NIST Special Publication 800-86). Gaithersburg, MD.

Okolica, J., & Peterson, G.L. (2010). Windows systems agnostic memory analysis. *Digital Investigation: The International Journal of Digital Forensics & Incident Response*, 7(1), S48-S56.

Okolica, J., & Peterson, G.L. (2011). *Extracting Forensic Artifacts From Windows O/S Memory.* Technical Report. Air Force Institute of Technology.

Osborne, G. (2012). *The Explore, Investigate, and Correlate Process for Information Visualisation in Digital Forensics.* PhD dissertation. University of South Australia.

Palmer, G. (2001). A Road Map for Digital Forensic Research. *First Digital Forensic Research Workshop (DFRWS)*. http://www.dfrws.org/2001/dfrws-rm-final.pdf

Russinovich, M., & Solomon, D. (2005). *Microsoft Windows Internals Fourth Edition*. Redmond, WA. Microsoft Press.

Scientific Working Group on Digital Evidence. (2011). *SWGDE and SWGIT Digital & Multimedia Evidence Glossary*. https://www.swgde.org/documents/Archived%20Documents/2011-01-14_SWGDE-SWGIT_Glossary_v2_4.pdf

Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. *Proceedings of the IEEE Symposium on Visual Languages*, 336-343.

Silberman, P., & C.H.A.O.S, (2005). FUTo. *Uninformed*. http://uninformed.org/?v=3&a=7&t=sumry

Stevenson, L., & Altholz, N. (2006). *Rootkits for DUMMIES*. Wiley Publishing, Inc.

Teerlink, S., & Erbacher, R. F. (2006). Improving the Computer Forensic Analysis Process Through Visualization. *Communications of the ACM*, 49(2), 71-75.

United States Computer Emergency Readiness Team. (2008). *Computer Forensics*. https://www.us-cert.gov/sites/default/files/publications/forensics.pdf

Volatility Development Team. (2013). Volatility: An Advanced Memory Forensics Framework. *Internet.* https://code.google.com/p/volatility/

Ware, C. (2004). *Information Visualization: Perception for Design (2nd ed.)*. San Francisco, CA. Elsevier.

Yadav, S. (2011). Analysis of Digital Forensic and Investigation. *VSRD International Journal of Computer Science & Information Technology* 1(3), 171-178.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
12-06-2014	Master's Thesis	September 2012 – June 2014

TITLE AND SUBTITLE

WINDOWS MEMORY FORENSIC DATA VISUALIZATION

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Baum, James B., Civilian, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-T-14-J-1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
Modern criminal investigators face an increasing number of computer-related crimes that require the application of digital forensic science. The major challenge facing digital forensics practitioners is the complicated task of acquiring an understanding of the digital data residing in electronic devices. Currently, this task requires significant experience and background to correctly aggregate the data their tools provide from the digital artifacts. Most of the tools available present their results in text files or tree lists. It is up to the practitioner to mentally capture a global understanding of the state of the device at the time of seizure and find the items of evidentiary interest. This research focuses on the application of Information Visualization techniques to improve the analysis of digital forensic evidence from Microsoft Windows memory captures. The visualization tool developed in this work presents both global and local views of the evidence based on user interactions with the graphics. The resulting visualizations provide the necessary details for verifying digital artifacts and assists in locating additional items of relevance. This proof-of-concept model can be modified to support various digital forensic target platforms including Mac OS X, Linux, and Android.

15. SUBJECT TERMS
Digital Forensics, Memory Analysis, Information Visualization, Windows XP, Interactive Tool, D3 JavaScript

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Peterson, Gilbert L., PhD, AFIT/ENG
U	U	U	UU	101	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4281 (gilbert.peterson@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18